Structural Assessment of Pile Supported Piers

by

Richard Jay Keiter

B.S., Mechanical Engineering
Purdue University, 1991

submitted to the Department of Civil and Environmental Engineering
and the Department of Ocean Engineering in Partial Fulfillment of the
Requirements for the Degrees of

Master of Science in Civil and Environmental Engineering

and

Master of Science in Ocean Engineering

at the
Massachusetts Institute of Technology
February 1999

© 1999 Richard J. Keiter.
All rights reserved.

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

Signature of Author Richard J. Keiter
Department of Civil and Environmental Engineering
January 15,1999

Certified by Jerome J. Connor
Jerome J. Connor
Professor of Civil and Environmental Engineering
Thesis Supervisor

Certified by J. Kim Vandiver
J. Kim Vandiver
Professor of Ocean Engineering
Thesis Reader

Accepted by A. J. Whittle
Andrew J. Whittle
Chairman, Departmental Committee on Graduate Studies
Department of Civil and Environmental Engineering

Accepted by J. Kim Vandiver
J. Kim Vandiver
Chairman, Departmental Committee on Graduate Studies
Department of Ocean Engineering

Structural Assessment of Pile Supported Piers

by

Richard Jay Keiter

Submitted to the
Department of Civil and Environmental Engineering
and the
Department of Ocean Engineering
on January 19, 1999 in Partial Fulfillment of the
Requirements for the Degrees of

Master of Science in Civil and Environmental Engineering

and

Master of Science in Ocean Engineering

19990406 100

Abstract

With the mobile nature of the armed forces, marine facilities are being encountered overseas for which no design data is readily available. To be able to consider such a pier in tactical planning, an assessment must be performed to estimate the load capacity of the pier. There is a group of technicians in the U.S. Navy that can perform rapid inspections on marine structures and gather data on the physical condition of the structure as well as the local environment. This data, combined with knowledge of design principles for waterfront structures, is used to provide a rapid estimate of the load capacity of the pier. This study focuses on a strategy for providing a rapid structural assessment of a waterfront pier given the information gathered during the on-site inspection combined with principles of waterfront design. The author has developed a program using the **C** programming language that, given the limited information gathered by Underwater Construction Team personnel, can be used in the field to provide an estimate of the structural capacity of an open, timber, pile-supported pier. The program prompts the user for various physical, environmental, and condition data and outputs various data files. A text file is produced which contains the inspection record that reflects the users input and the assessment results for the pier being analyzed. A MATLAB® script file is produced which can be used for subsequent processing.

Thesis Supervisor: Jerome J. Connor
Title: Professor of Civil and Environmental Engineering

Thesis Reader: J. Kim Vandiver
Title: Professor of Ocean Engineering

DTIC QUALITY INSPECTED 3

Abstract

With the mobile nature of the armed forces, marine facilities are being encountered overseas for which no design data is readily available. To be able to consider such a pier in tactical planning, an assessment must be performed to estimate the load capacity of the pier. There is a group of technicians in the U.S. Navy that can perform rapid inspections on marine structures and gather data on the physical condition of the structure as well as the local environment. This data, combined with knowledge of design principles for waterfront structures, is used to provide a rapid estimate of the load capacity of the pier. This study focuses on a strategy for providing a rapid structural assessment of a waterfront pier given the information gathered during the on-site inspection combined with principles of waterfront structure design. The author has developed a program using the **C** programming language that, given the limited information gathered by UCT personnel, can be used in the field to provide an estimate of the structural capacity of an open, timber, pile-supported pier . The program prompts the user for various physical, environmental, and condition data and outputs various data files. A text file is produced which contains the inspection record that reflects the users input and the assessment results for the pier being analyzed. A MATLAB® script file is produced which can be used for subsequent processing.

Acknowledgments

There are those whom without their assistance this work could not have been accomplished. Mr. Stan Black of the Naval Facilities Engineering Service Center provided a tremendous amount of assistance. If the reference existed, Stan knew where to find it. Professor Kim Vandiver of the Ocean Engineering Department at the Massachusetts Institute of Technology (MIT) who took the time to read this work and does an outstanding job of teaching...in the classroom and out. And finally, Professor Jerome Connor of the Civil and Environmental Engineering Department at the MIT, who has served not only as thesis advisor, but also as faculty advisor for my short stay here at MIT. Professor Connor listened and helped to smooth out the bumps.

Table of Contents

Abstract 2
Acknowledgments 3
Table of Figures 6
List of Tables 6
I. Introduction 7
A. General 7
B. Underwater Construction Teams 8
C. Scope of this Study 8
II. Pier Configuration and Nomenclature 10
A. Pier Construction 11
1. Piles *11*
2. Decking *12*
III. Design Considerations 14
A. Material 14
B. Loading 15
1. Vertical *16*
2. Lateral *17*
3. Dynamic *19*
C. Seismic 19
D. Geotechnical 20
E. Ice 20
F. Factor of Safety 20
G. Miscellaneous 21
IV. Underwater Construction Team Inspection Data 22
A. Levels of Inspection 22
B. Pier Inspection Documentation 22
1. Physical Dimensions *23*
2. Environmental Data *24*
V. Assessment 26
A. Loads 26
1. Dead *26*
2. Wind *28*
3. Current *31*
4. Waves *31*
5. Dynamic *36*
B. Piles 38
1. Fixity *38*
2. Vertical *39*
3. Lateral *41*
C. Decking 43
1. Stringers - Simply Supported Beams *43*

2. Continuous Beams *49*
VI. Rapid Structural Assessment- Pier **55**
VII. Conclusion **56**
References **57**
Appendix A - Levels of Inspection **58**
Appendix B - Pile Inspection Record **59**
Appendix C - Pile Condition Ratings for Timber Piles **60**
Appendix D - Crane Loading Data Charts **61**
Appendix E - Program Listing **67**
Appendix F - Sample RSAP Output Data File **125**

Table of Figures

Figure 1. Examples of finger piers (top view) 7
Figure 2. Typical open, pile supported pier 10
Figure 3. Typical timber pier structure and nomenclature 11
Figure 4. Example of a simple fender 12
Figure 5. Pile head connection details 13
Figure 6. Decking detail 13
Figure 7. Damage profiles for the woodgribble (left) and the teredo (right) 15
Figure 8. Typical loading on a marine structure. 16
Figure 9. Displacement to wind load relation (Tsinker[1]) 30
Figure 10. Depth to fixity illustrated. 39
Figure 11. Bending modes and effective lengths of 2 columns. 40
Figure 12. AASHTO Truck specifications 45
Figure 13. Comparison of moment equation results vs. *L/a* - HS Loading 46
Figure 14. H Loading resulting moments vs. *L/a* 48
Figure 15. Simplified Three-Moment Equation Terms[13] 50
Figure 16. Three-moment equation model output 51
Figure 17. Shear-Moment diagram: Pile spacing - 6'; Axle - centered 51
Figure 18. Shear-Moment diagram: Pile spacing - 10'; Axle - centered 52
Figure 19. Sample MATLAB Output 55

List of Tables

Table 1. Properties of timber commonly used in marine construction[3] 14
Table 2. Level of inspection versus detectable damage to timber waterfront structures 22
Table 3. Coefficients C_1 and C_2 for wind force calculation. 29
Table 4. Hydrodynamic coefficients, C_D and C_M 34
Table 5. Soil compatability table for D_f 39
Table 6. Forklift wheel loads and dimensions. 48

I. Introduction

A. General

Maritime transportation has generally been the most convenient and least expensive means of transporting goods[1]. As technology in the ship design and construction industry has improved, cargo ships have become larger and more specialized. Accordingly, complex port facilities worldwide have been developed to accommodate waterborne cargo. These marine facilities typically include piers, wharves, quays, and dolphins as well as a wide array of cargo handling equipment such as forklifts, cranes, and stacking straddle carriers. Historically, the finger pier (see Figure 1) was the most characteristic type of berth construction [1]. Though modern construction has been trending towards more use of concrete, steel, composites, and combinations, timber has been, and continues to be, a primary construction material.

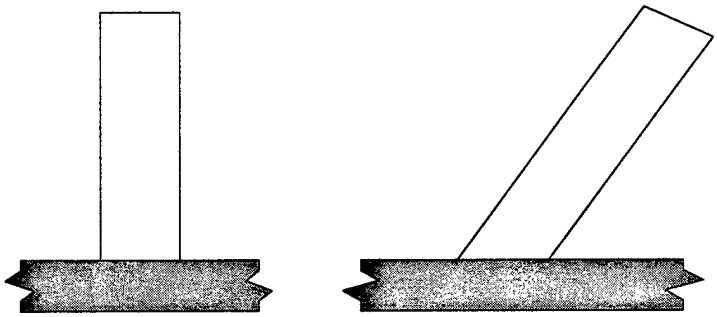


Figure 1. Examples of finger piers (top view)

There are a number of timber, finger piers still in service in the United States and overseas. For many of the marine facilities located in the U.S., adequate design information is available that, with current condition data, can be used to determine the load capacity of the structure. However, with the mobile nature of the armed forces, marine facilities are being encountered overseas for which no design data is readily available. To be able to consider such a pier in tactical planning, an assessment must be performed to estimate the load capacity of the pier. There is a group of technicians in the U.S. Navy that can perform rapid inspections on marine structures and gather data on the physical condition of the structure as well as the local environment. This data, combined with knowledge of design principles for waterfront structures, can be used to provide a rapid estimate of the load capacity of the pier.

B. Underwater Construction Teams

The Underwater Construction Teams' (UCTs) mission tasks them with: "Providing(sic) a capability for construction, inspection, repair, and maintenance of ocean facilities in support of Naval and Marine Corps operations..." and "Maintaining(sic) [the] capability to support a Fleet Marine Force (FMF) amphibious assault..." To accomplish their assigned mission, there are a number of capabilities which the UCTs must maintain. Among these, the items of interest to this study include[2]:

- During the initial period of contingency mobilization, provide underwater construction support of Naval Beach Groups, Harbor Defense Groups, and other fleet units as directed.
- Construct, inspect, and repair ocean facilities in support of Naval and Marine Corps operations in the combat zone or at forward area support bases.
- Respond to emergency inspection and repair of essential fleet water-front systems within 48 hours.

As these capabilities indicate, and anecdotal evidence supports, the UCTs must be able to provide rapid inspections of waterfront facilities which are being considered for use in tactical operations. Specifically, the UCTs perform waterfront inspections of ocean facilities in support of combat operations. While the results of the inspection may reveal that the pier is sound and undamaged there is currently no method of taking the information gathered during an inspection and quickly estimating the structural capacity of the pier in question.

C. Scope of this Study

This study focuses on a strategy for providing a rapid structural assessment of a waterfront pier given the information gathered during the on-site UCT inspection combined with principles of waterfront structure design. The author has developed a program using the **C** programming language that, given the limited information gathered by UCT personnel, can be used in the field to provide an estimate of the structural capacity of an open, timber, pile-supported pier . The program prompts the user for various physical, environmental, and condition data and outputs various data files. A text file is produced which contains the

inspection record that reflects the users input and the assessment results for the pier being analyzed. A MATLAB® script file is produced which can be used for subsequent processing.

II. Pier Configuration and Nomenclature

The general designation for the place where a vessel can be moored is a *dock*. A *pier* is a dock that extends outward, perpendicular to, or at some skew angle to, the shoreline. A pier is essentially a free-standing structure, shore connected at one end, which allows berthing of vessels along both sides. The most common pier construction consists of open, pile supported structures which include a decking system constructed on a pile foundation. The foundation contains a series of evenly spaced pile groups, or *bents*, as shown in Figure 2. The pile bents may be further strengthened to resist lateral loads by the addition of *batter* piles or by being rigidly cross-braced.

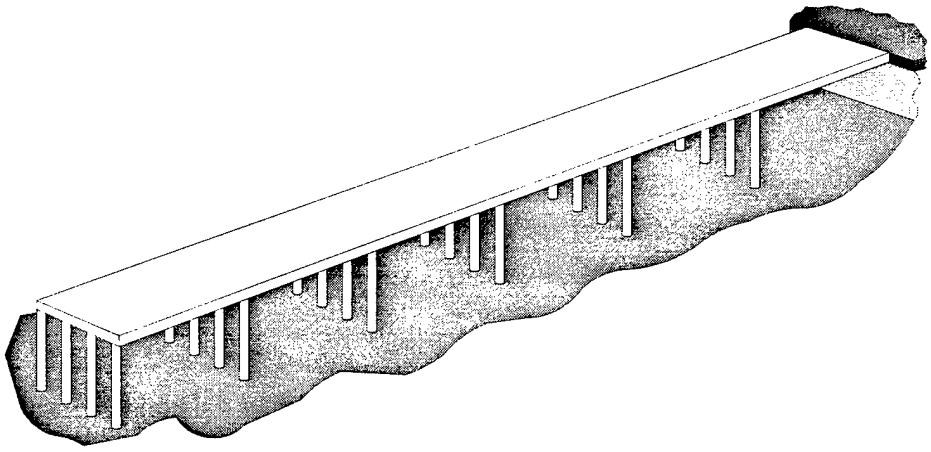


Figure 2. Typical open, pile supported pier

Timber has been the traditional material for waterfront construction. It is durable and it possesses good impact resistance and the ability to distribute loads effectively[3]. It is particularly durable in locations which are free from biological organisms and subjected to continuous "wet" conditions. Tsinker(1) sites the example of the more than 100 year old Brooklyn Bridge which is supported on a timber cribbage foundation. Currently, all-timber pier construction is usually relegated to lightly loaded sites such as small craft harbors and public facilities.

There are many types of timber used in marine construction. For pilings, the type selected generally depends upon availability and cost. Usually, piling timber is treated with chemical agents such as creosote to deter waterborne biological organisms, such as *limnoria* or *teredos*, and prolong the life of the pile. Decking timber is generally a hardwood such as white oak but

may vary based upon availability. It is not necessary to treat the decking timber since it is well away from the splash zone and not subject to the aforementioned biological hazards.

A. Pier Construction

Timber pier construction is generally of the type shown in Figure 3. The shaded components may be present but are not required.

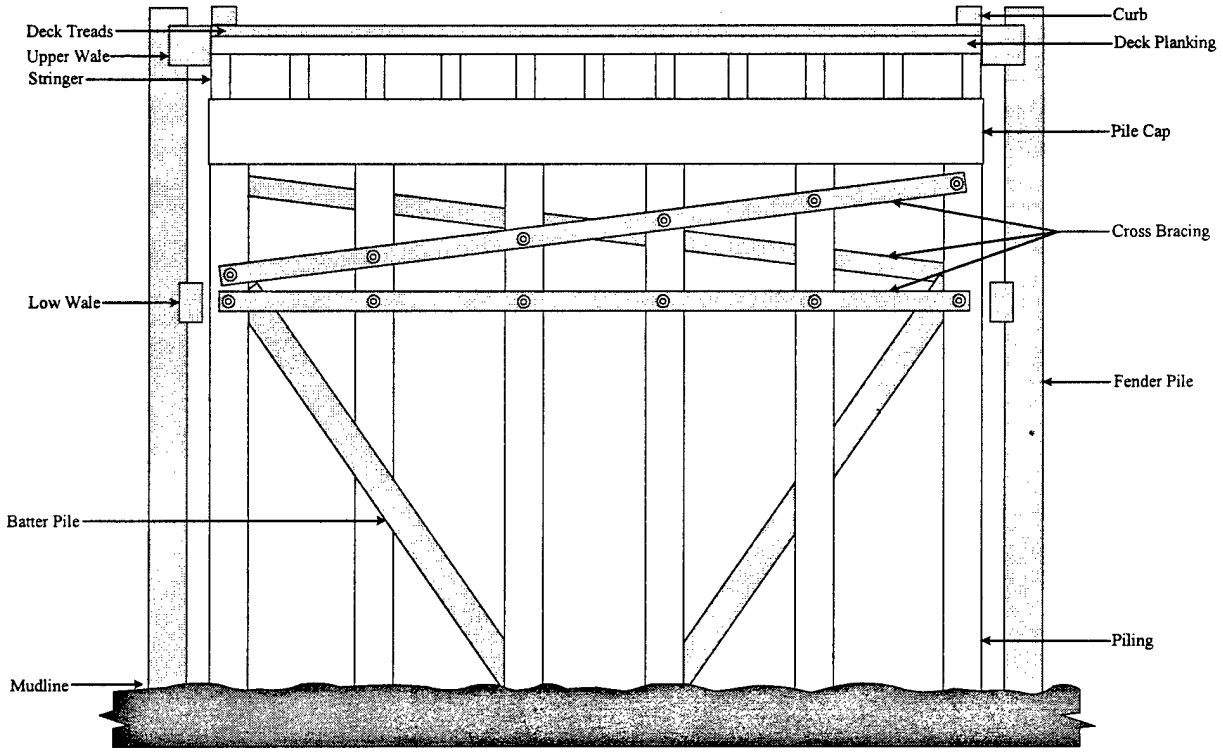


Figure 3. Typical timber pier structure and nomenclature

1. Piles

There are three types of piles that contribute to a piers ability to withstand loading. *Bearing* piles are vertical piles that support the vertical load of the pier and may provide lateral support as well. Bearing piles are friction-type, end-bearing, or a combination of both. *Batter* piles are placed on an angle to provide lateral support. Additionally, they may provide vertical support as well. As with bearing piles, batter piles may be friction-type, end-bearing or a combination of both. For batter piles that are a combination of both, a batter may contribute to the lateral

resistance in either compression or tension. Conversely, end-bearing batter piles may contribute only if in compression. Lastly, there are *fender* piles. There are many configurations of fender system comprised of piles with the simplest shown in Figure 4. A fender system is installed to absorb the energy imparted to the pier while a ship is berthing, thus decreasing the lateral displacement of the pier and ultimately reducing the loads on the pier. Fender piles are generally considered sacrificial in nature and require regular maintenance to minimize damage from docking impact to the pier.

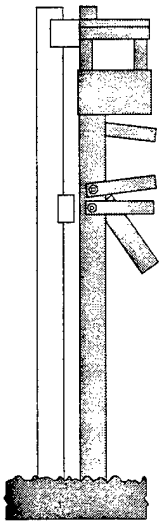


Figure 4. Example of a simple fender

2. Decking

Decking consists of everything above the pile ends. This includes the pile *caps*, deck *stringers*, deck *planking*, and deck facing. Of these, the deck facing is the only component that does not contribute to the structural capacity of the pier. Its purpose is to protect the decking against damage from vehicular traffic, etc. The decking is usually placed well above the splash zone. Pile caps, shown in Figure 5[3], consist of either a solid beam that spans across the tops of the pilings

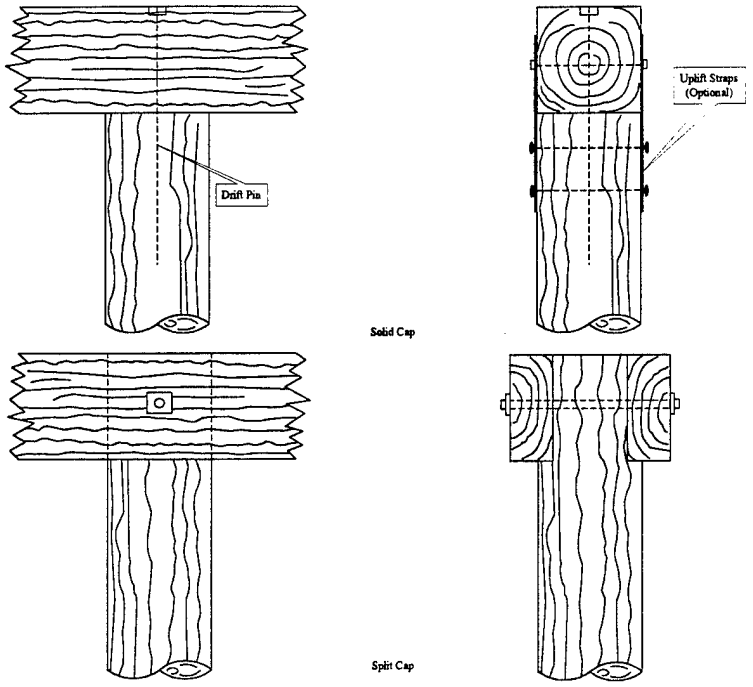


Figure 5. Pile head connection details

in the bent or two beams that are situated on either side of the pile tops. In both cases, the pile head is considered a pinned connection. Deck stringers are evenly spaced timbers, placed edgewise, that span the bents. Lastly, the deck planking spans the stringers to complete the load carrying structure. Figure 6, adapted from NAVFAC[4], details these various components.

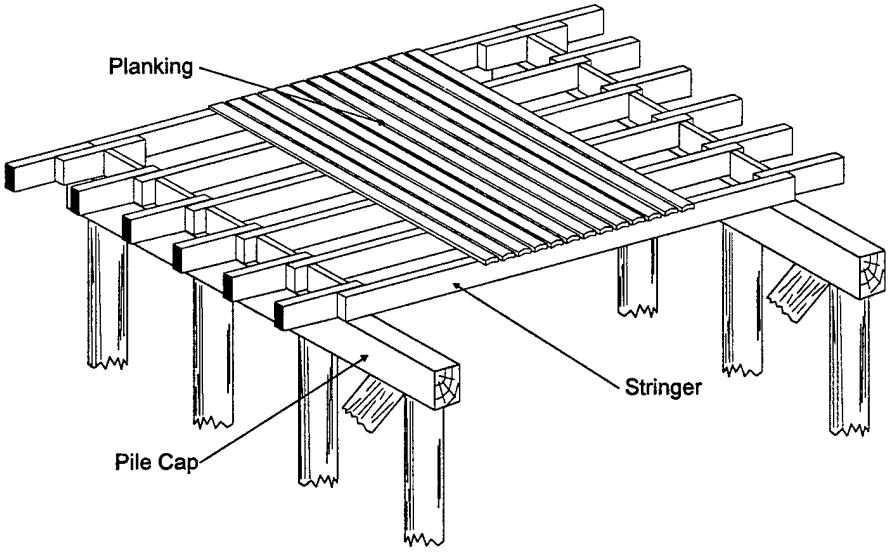


Figure 6. Decking detail

III. Design Considerations

When designing a marine structure, there are no definitive, binding building codes or standards to which the designer may refer. However, "...there are several guideline codes of practice to which the designer may refer for general design and for specific requirements."[3] Before a meaningful discussion of the analytic methods used in conducting the structural assessment can be properly conducted, it is important to understand the various considerations that are an integral part of a marine facility design.

A. Material

There are various types of timber used throughout waterfront construction. Timber is used because it is durable, convenient to work with, possesses good impact resistance and the ability to distribute loads effectively[3]. Table 1 contains properties of a few types of timber commonly used for pilings. Often, the softer timbers will be pressure-treated before use. For

Timber Type	Elastic modulus in bending (psi)	Proportional limit in compression parallel to grain (psi)	Proportional limit in compression perpendicular to grain (psi)	Shearing strength parallel to grain (psi)	Unit weight (pcf)
Douglas Fir (coast type)	1,950,000	5,850	870	1,160	34±
Southern Yellow Pine (long leaf)	1,990,000	6,150	1,190	1,500	36±
Greenheart (Ocotea radiaei)	3,700,000	10,140	2,090	830	66±
Azobe (Ekki) (Lophira procera)	3,000,000	10,260	2,870	2,650	65.5

*These values for "air-dry" wood...typically around 12% moisture. For wood with a higher moisture content, such as wood that is continuously submerged, strength properties are reduced and unit weights increased

Table 1. Properties of timber commonly used in marine construction[3]

the decking structure of a pier, hardwood, such as white oak, is often used. The main threats to timber marine structures are rot, mechanical damage, or marine organism attack. Rot is caused primarily by stagnant fresh water. When present, rot is usually found in the structural components above the pilings and can be difficult to detect. Mechanical damage can be caused by any number of sources including berthing operations and cargo handling. It can be found in the decking and the pilings. However, major mechanical damage to pilings is usually

confined to those pilings located at the perimeter of the structure. It is in those locations that the pilings can come into direct contact with ships, barges, tugs, etc. In the interior of the piling group, mechanical damage is caused primarily by abrasion and wear from floating debris. Of the main threats to timber marine structures, marine organism attack is the most prevalent. There are two prominent types of marine borers: the woodgribble of the Limnoria family and the teredo, which is a mollusk.

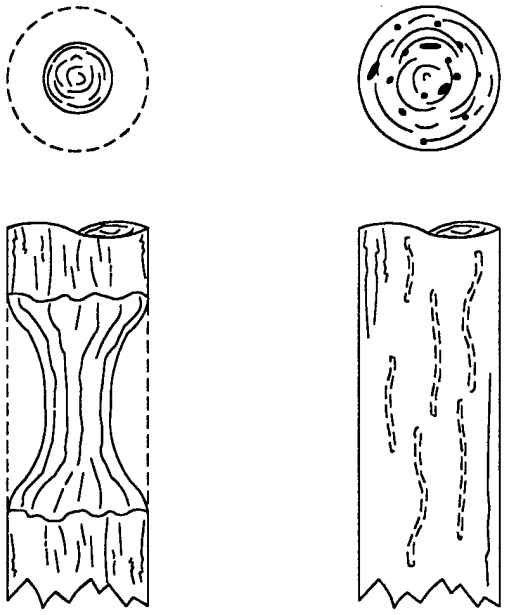


Figure 7. Damage profiles for the woodgribble (left) and the teredo (right)

The woodgribble eats away shallow furrows at the piling surface in the surf zone leaving an "hourglass" appearance. The teredo tunnels throughout the pile leaving the pile riddled with holes. Examples of this damage can be seen in Figure 7. Because most of the teredo damage is inside the pile, it takes a more experienced eye to detect it.

B. Loading

"Design of fixed piers and wharves is usually controlled by live load and lateral load requirements."[5] Various loads must be considered when assessing the structural capacity of a pier. These loads fall into one of three general categories of loading: *permanent* load which is also known as dead load and is a vertical loading; *temporary* loads which include live loads from operations and environmental loads and contribute to both vertical and lateral loading; and *special* loads which include accidental loads, seismic loads and other unusual loading. A

structure is not always loaded as designed and, thus, when designing a marine structure, "...the selection of design loads is a problem of statistics and assessment of probability."[1] Figure 8 illustrates the numerous sources of loading that a pier may experience.

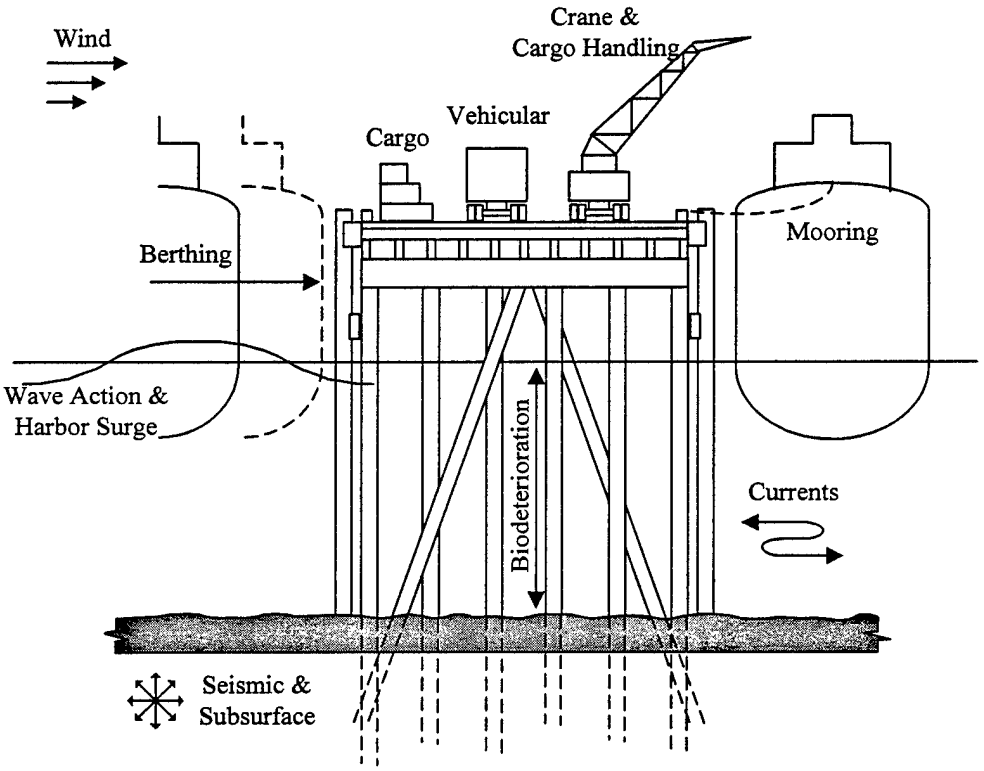


Figure 8. Typical loading on a marine structure.

Compared to other types of structures, piers are typically designed to support relatively heavy transient loads as well as a relatively large lateral load. The design vertical load capacity is generally governed by deck and cargo live loading, vehicle loading, and mobile equipment used on the pier. The design lateral load is governed by berthing and mooring forces. Loading design considerations are discussed as to the contribution they make to the vertical and lateral components of loading.

1. Vertical

Vertical loading includes the dead load, which is the weight of the structure and everything permanently attached such as any mooring hardware, curbs, light poles, etc. The vertical load also contains live load contributions which consist of uniform loading and point loading from cargo, vehicular traffic, and material handling equipment such as forklifts and mobile cranes,

which are rubber tire or crawler tracked mounted. When designing a pier, there are two concepts employed in the formulation of design loads: the "real-life" load assumption based upon miscellaneous loads falling in a line or concentrated load category, and the "equivalent uniform" load assumption[1]. The latter can be misleading. For example, a pallet or container may be assumed to provide uniform loading on the order of 200-600 psf. However, a pallet or container may actually be loaded in such a way that there is concentrated loading that exceeds the assumed uniform loading. Thus, it is best to compromise with a combination of both concepts. When looking at the influence of loading, concentrated loads dominate at the decking while uniform loading tends to dominate the substructure such as piling size.

2. Lateral

The lateral loads consist primarily of mooring forces, berthing forces, and environmental loading. The mooring forces are usually from environmental loading on the ship alongside the pier. The berthing forces are from the actual berthing operations where there are potential impacts incident upon the pier from the ship. This assessment deals with the environmental aspects of lateral loading since the berthing forces are highly unpredictable, varying with ship size, speed, angle of approach, and fender system. It is assumed that if a pier is of importance tactically, great care will be taken to see that damage, such as the type experienced during careless berthing, will be avoided.

a. Wind

Wind contributes primarily to the lateral loading on a pier. It blows from many directions and can change without notice. The wind impinging upon a surface, increases the pressure on that surface and results in a force loading. However, given the construction characteristics of an open pier, the loading on the structure itself is minor compared with the wind effects of the ship moored along side. The exposed, directional, surface area of the ship is susceptible to wind loading which is then transferred to the pier. When designing a pier, historical wind data, along with the design ship size, is analyzed to size the structural members according to the predominant wind direction. Also, it is assumed that under high wind or wave conditions, vessels will leave the berth and crane operations will cease so that there is a limiting design wind. The wind speed used in loading calculations is the wind 10 m above the surface of the

water. If the wind speed is measured at a different height, a relation is available to convert it to a 10 m equivalent. The maximum wind load on a pier will be when the wind direction is perpendicular to the pier.

b. Current

"Current forces are normally neglected in the design of harbor structures. However, the rational design of exposed piling as a column...requires that lateral forces due to current be considered."[6] Currents can be caused by the wind, river flow, and tide flow. The current speed is usually maximum at the surface and reduces to zero at the bottom. It is possible to have opposing sources such as might be seen when a river current flows in on direction and the wind induces a current flow in the opposite direction. If strong enough, a current can increase the pressure head on one side of a moored vessel causing a considerable increase in mooring forces. The submerged, directional, surface area of the ship is susceptible to current loading which is then transferred to the pier. As with the wind effects, current effects are present on the pilings of the pier, but the effects from the ship dominate. The maximum current load on a pier will occur when the current direction is perpendicular to the pier.

c. Waves

As with wind, wave design considerations rely on historical data. This data is usually presented as short-term data, which is presented in terms of occurrence frequency and yearly averages for each month, and long-term wave statistics, which are usually given in terms of maximum wave height versus statistical return period. If considering unidirectional sea waves, they are usually represented as a wave spectrum due to their irregularity. When expanding to the more realistic case where waves are multi-directional, there have been three dimensional wave spectra developed but use in regards to harbor design/analysis is limited[3]. Waves are generally classified as one of the following: wind generated waves; ship generated waves; astronomical tidal waves; storm surges; harbor seiches, which is the excitation of a harbor due to long period ocean waves; tsunamis; capillary waves; and interval waves[1]. Of these, wind is the primary cause of waves. Therefore, wind waves and their associated swells are considered in the design of coastal facilities. Given the oscillatory nature of waves, wave loads are dynamic. However, for the range of water depths encountered with coastal piers,

wave load can generally be represented as equivalent static loads[3]. The manner in which wave loads and their associated calculations are handled depends upon the member or structure dimension relative to wavelength. If the member being investigated is a pile and its diameter is small compared to the wavelength, the wave is not influenced by the pile and the resulting force on the pile is due to water particle velocities and accelerations. These are also know as *drag* forces and *inertial* forces respectively. If the member or structure being investigated is large enough to affect the passing wave, diffraction and wave scattering must be considered. If the structure is very wide, such as a ship, then reflection occurs and the forces are treated as a rise in hydrostatic pressure head. Gaythwaite[3] provides the following criteria for application of wave force calculations:

- For $D/\lambda \leq 0.2$, drag and inertia forces dominate; use the Morison equation.
- For $D/\lambda > 0.2$, diffraction effects become increasingly important; use diffraction theory.
- For $D/\lambda > 1.0$, pure reflection conditions exist; treat the structure as a seawall.

Open piled structures are the preferred type of construction at locations exposed to heavy waves because they enable practically free passage of waves[1]. The maximum wave loading occurs when the wave propagation direction is perpendicular to a moored ship.

3. Dynamic

The two predominant contribution to dynamic loading are periodic waves and seismic activity. If a wave having a period, T_w, close to that of the moored ship is incident to the ship, it may result in resonant phenomena that will amplify the ships mooring forces on the pier. Determination of these effect require extensive analysis of the location, to include scale modeling to calibrate the mathematical model, and detailed seismic history data.

C. Seismic

Typically, marine structures, such as piers, are designed for high lateral loading and thus, are relatively rigid with natural periods on the order of 0.5 seconds or less[3]. Additionally, immersion in water provides damping in addition to that inherent in land-based structures. However, there may be exceptions such as piers having vertical cantilever piles. For nearshore pier structures, a check for seismic forces should be conducted in accordance with

specifications provided by The American Association of State Highway Transportation Officials or AASHTO[7].

D. Geotechnical

Soil properties contribute greatly to the ability of a pile to withstand vertical loading either in end bearing capacity, friction bearing capacity or a combination of both. When designing a pier, extensive testing is conducted on the soil foundation to determine its ability to support loading. This information is used to determine the depth to which the piles must be driven and the point of *fixity*, D_f, which is the point at which the pile is considered fixed in the soil. The fixity is used to determine the unsupported pile length in load capacity calculations. Since the UCT will not have access to design soil information in a tactical situation, they must either test the soil to determine soil type or rely upon a basic assumption. The current method[8] for dealing with unknown piers is to assume that the constructors used piles sized such that the soil and geotechnical conditions exceed the strength of the pile. Thus the pile strength is the limiting factor when considering the foundation strength.

E. Ice

If present, ice can exert a lateral force upon the pilings at the waterline. If the ice is sheeted, both the wind and the current can exert shear force on the ice. Also, the ice can impose vertical loading on the piling with the tide change. However, if ice thick enough to cause significant loading on the pier is present, it is unlikely that a ship would be able to moor there. Tsinker [1] states that, normally, conventional open piled structures are not feasible in heavily-ice-infested waters.

F. Factor of Safety

When designing a waterfront structure for loading, great care must be taken to ensure that the structural member will not fail under design loading. When using AASHTO[7] specifications for pier design, as recommended in much of the literature, allowable stresses, with reduction factors for various conditions, are provided for the building material used. Often, these values can be converted directly to allowable loads. For pilings, however, an additional check for critical loading must be performed since the piling behaves as a column and is subject to

buckling. Thus, the factor of safety is used to reduce the critical load to an allowable load that may, for longer piles, be less than the allowable load determined using allowable stress in compression parallel to the grain.

G. Miscellaneous

-When performing a design analysis, settlement of foundations and the resulting effect on pile cap load capacity reviewed. For this assessment, however, settlement will not be considered unless it is severe. And then only by the personnel on-site.

-Often times there are connectors in a marine structure. It is left to the inspection team to assess the condition of any connectors that might be present and determine the impact to the structural capacity. If the connectors holding cross-bracing in place are severely deteriorated, it would be best to model the pier as having no cross bracing. While this may limit the lateral capacity of the pier, it will prevent an overly optimistic capacity assessment.

-Biological fouling such as mollusk growth can add to the drag of the pilings when considering current and wave effects. It can also add weight to the structure at low tide since the growth will be above the waterline.

IV. Underwater Construction Team Inspection Data

"A major portion of the UCT's activity is directly related to the underwater inspection of a wide variety of waterfront structures and other marine facilities. UCT underwater inspections are primarily visual observations of the facility being inspected."[9] This quote comes directly from the NAVFAC P-990: Underwater Construction and Repair Techniques manual and highlights the fact that waterfront structure inspections are a UCT core competency. The P-990 provides the UCTs with the guidance to perform waterfront inspections on many types of structures including those constructed of timber. Additionally, emphasis is placed upon the importance of reliable, detailed inspection documentation for subsequent engineering assessment.

A. Levels of Inspection

The P-990 defines three levels of inspection which provide increasingly more detail as one moves from the basic Level I inspection to the more advanced Level III inspection. A level III inspection is still primarily a visual inspection but will often include some Non-Destructive

Level	Purpose	Detectable Defects
I	General visual to confirm as-built condition and detect severe damage	-Major losses of wood -Broken piles and bracings -Severe abrasion or marine borer attack
II	Detect surface defects normally obscured by marine growth	-External pile damage due to marine borers -Splintered piles -Loss of bolts and fasteners -Early borer and insect infestation
III	Detect hidden and imminent damage	-Internal damage due to marine borers (internal voids) -Decrease in material strength

Table 2. Level of inspection versus detectable damage to timber waterfront structures

Testing (NDT) techniques. In some cases, partially destructive techniques, such as core sampling, are used[9]. Appendix A provides a more detailed description of each level of inspection as described in the P-990[9]. However, a summary of the purpose and detectable defects for each level of inspection are shown in Table 2.

B. Pier Inspection Documentation

As stated previously, it is important that an inspection produce reliable, detailed documentation. The P-990 provides a standard form for the reporting of pile condition as

observed during a UCT pier inspection. This form, the Pile Inspection Record (see Appendix B), allows for each pile to be classified according to one of five condition codes: ND (no damage); MN (minor damage); MD (moderate damage); MJ (major damage); and SV (severe damage). Appendix C provides an explanation for each of these codes and gives representative diagrams of each pile condition. The Pile Inspection Record also allows for recording of the type of damage to the pile being inspected. This is important when evidence of biological damage, such as that from marine borers, exists. Additional information, which is easily collectible, required for the purpose of this paper include physical dimensions of various structural members and local environmental data such as wind, wave and current.

1. Physical Dimensions

Dimensions of the various structural components of the pier are required to assess their mechanical strength. These include measurements which are discussed individually.

a. Pile Diameter and Depth

Typically, pile diameter, D, is consistent throughout the pier structure and will be treated as such for the purposes of this assessment. However, care should be given when measuring the diameter of timber piles since they are often tapered and may experience a reduction in diameter from the pile cap to the embedded portion of the pile. Thus, measurement should be taken near the bottom to ensure an accurate calculation of the pile strength. Allowance has been made in the program for the possibility that batter pilings are of a different diameter than the bearing piles. In an extreme case, such as a damaged pile having been replaced, a different pile diameter may be entered for a bearing pile. This will be discussed further in the Assessment section of this paper. Also of interest are the depth, d, and water-surface-to-point-of-connection dimensions. These values combine to provide the unsupported length, L_u, of the pile.

b. Pile Cap, Stringer, and Decking Dimensions

These components of the load bearing structure can all be modeled as beams. The height and width of these members are required to perform the assessment. Care must be given to record

the vertical dimension as "height", h, and the horizontal dimension as "width", b, so that the proper modulus in bending may be calculated.

2. Environmental Data

When designing a waterfront structure such as a pier, detailed environmental data is collected and used in the analysis to ensure that the resultant structure can withstand the environmental loading expected given the construction budget. This information will include soil borings, temperature ranges, current data, and statistical wind data such as a *wind rose*. A wind rose graphically represents the direction, frequency, and intensity of the average winds at a particular location over a period of time.[1] When conducting inspections at known facilities, the UCT inspection team may have access to this historical data. However, when conducting an inspection in a tactical environment, the UCT inspection team will have little, if any, of this data available to them. They must collect it via observation.

a. Wind

Prevailing and extreme wind speeds and directions and their frequency of occurrence are of primary concern when considering wind loading[3]. The UCT inspection team can collect data from the time of their arrival on site to estimate the average wind speeds and prevailing directions from which they come. They are not capable of determining extreme wind speeds associated with long return periods such as those experienced in 50- or 100-year storms. However, given that the situation is tactical and that a ship interested in using the pier for off-loading will have access to meteorological information, the wind speed can be monitored for extreme conditions. The information of interest regarding wind is the velocity magnitude, V_w, the direction it is blowing relative to the longitudinal axis of the pier, θ_w, and the height above the water surface, h_w, that the measurement was taken/observed.

b. Current

The current is usually estimated by divers in the water. In most cases the current will run parallel to the shoreline and the current velocity will decrease with depth. The information of interest regarding current is the velocity magnitude, V_c, and the direction it is flowing relative to the piers longitudinal axis θ_c. It is best to measure the current at the surface of the water.

c. Waves

There are two types of waves of interest to the loading on waterfront structures: sea waves and waves caused by passing vessels. It may be difficult to distinguish the various components of the incident waves. Therefore, it is best if UCT personnel measure the maximum periodic wave height, H_{wave}, measured peak to trough, period, T_{wave}, and direction, θ_{wave}. Care must be take to discount single occurrence waves in this observation.

d. Soil

The capability to perform geotechnical testing using hand powered tools has been recently introduced into the UCTs. However, during conversations with personnel from the Naval Engineering Service Center, Port Hueneme, CA, additional experience is required with these tools before reliable soil strength information can be obtained. Information regarding soil type may be of use for determining depth to fixity, D_f.

V. Assessment

In this section, the methods for determining the forces and load capacity are developed. The general approach taken is to assess only the structural components of the pier. If additional structures are present on the pier or there is installed equipment present on the pier, this must be accounted for on-site in accordance with good engineering practice. In all cases, closed end solutions equations are used since they readily lend themselves to computer programming techniques. Gaythwaite[3] suggests that this is not unusual for near-shore waterfront structures.

A. Loads

1. Dead

Ideally, when calculating the deadweight of the structure, detailed measurement would be taken of every pier member and attachment such as those used in mooring. However, given the nature of the situation in which the UCTs will be conducting such an inspection, this approach is unreasonable. Therefore, a simplifying approach is needed which will err on the conservative side. Before continuing with the actual weight calculations, a few simple dimension calculations must be made from the input provided by the inspection team.

$$l_{pier} = (\#\,of\ bents)(bent\ spacing) \qquad \textbf{Eq. 1}$$

$$w_{pier} = \left(\frac{piles}{bent} - 1\right)(pile\ spacing) \qquad \textbf{Eq. 2}$$

where:

l_{pier}	=	length of pier, ft,
$\#\ of\ bents$	=	total number of bents,
$bent\ spacing$	=	center to center, ft,
w_{pier}	=	width of pier, ft,
$piles/bent$	=	# of bearing piles in each bent,
$pile\ spacing$	=	distance center to center of piles in a bent, ft.

When considering the piles contribution to deadweight, an allowance must be made for buoyancy effects will act to reduce the weight of the pile. Thus, the weight of the pile will be calculated as follows:

$$wt_{pile,n} = \frac{\pi D^2}{4}\left(L_n \rho_{pile} - d_n \rho_{water}\right) \qquad \textbf{Eq. 3}$$

where:

$wt_{pile,n}$	=	weight of pile in bent n, lb,
n	=	bent number,
D	=	diameter of pile, ft,
L_n	=	length of piles in bent n from mudline to pile cap, ft,
ρ_{pile}	=	density of pile material, lbm/ft^3,
d_n	=	depth at bent n, ft,
ρ_{water}	=	density of seawater, 64 lbm/ft^3.

The weight of the pile cap is given as follows:

$$wt_{pile\,cap} = b_{pile\,cap} h_{pile\,cap} w_{pier} \rho_{decking} \qquad \textbf{Eq. 4}$$

where:

$wt_{pile\,cap}$	=	weight of pile cap, lb
$b_{pile\,cap}$	=	base dimension, ft,
$h_{pile\,cap}$	=	height dimension, ft,
$\rho_{decking}$	=	density of pile material, lbm/ft^3,
w_{pier}	=	pier width.

The weight of the deck planking per bent is given as follows:

$$wt_{planking} = h_{planking} w_{pier} \left(bent\ spacing\right) \rho_{decking} \qquad \textbf{Eq. 5}$$

where:

$wt_{planking}$	=	weight of planking, lb
$h_{planking}$	=	height dimension of planking, ft,
w_{pier}	=	pier width, ft,
$bent\ spacing$	=	center to center spacing, ft,
$\rho_{decking}$	=	density of decking material, lbm/ft^3.

The weight of the deck stringers per bent is given as follows:

$$wt_{stringers} = b_{stringer} h_{stringer} \left(bent\ spacing + b_{pile\,cap}\right)\left(\frac{w_{pier}}{stringer\ spacing} + 1\right)\rho_{decking} \qquad \textbf{Eq. 6}$$

where:

$wt_{stringer}$	=	stringer weight per bent, lb,
$b_{stringer}$	=	stringer base dimension, ft,
$h_{stringer}$	=	stringer height dimension, ft,
$bent\ spacing$	=	center to center spacing, ft,
$b_{pile\,cap}$	=	pile cap base dimension, ft,
w_{pier}	=	pier width, ft,
$stringer\ spacing$	=	distance center to center of stringers, ft,
$\rho_{decking}$	=	density of decking, lbm/ft^3.

To account for the various additional members of the pier, such as mooring accessories, whalers, curbing, etc. (excluding any buildings present on the pier), the weight of the decking will be increased by 15%. Thus, the weight carried by each bent can be expressed as follows:

$$wt_{bent} = 1.15\left(wt_{pile\,cap} + wt_{planking} + wt_{stringers}\right) \qquad \textbf{Eq. 7}$$

The weight can then be distributed as follows:

$$load_{pile,dead,\text{int},n} = \frac{wt_{bent}}{(\#\,piles\ per\ bent - 1)} + wt_{pile,n} \qquad \textbf{Eq. 8}$$

for an interior pile. The exterior piles support only half as much deck area as do the interior piles. However, since a majority of the additional weight is carried by the external piles the weight for those piles will be adjusted even further and is represented by:

$$load_{pile,dead,ext,n} = \frac{wt_{bent}}{2(\#\,piles\ per\ bent - 1)} + \frac{0.15wt_{bent}}{2} + wt_{pile,n} \qquad \textbf{Eq. 8a}$$

2. Wind

When calculating wind pressure, it is necessary to do so using the *sustained wind velocity* which is defined as the wind speed averaged over one minute. Also, the wind speed generally used for evaluation of wind loading is the sustained wind speed at located 10m above the water surface. If the wind speed observation is taken at another height, a correction must be applied to the measured, sustained wind velocity to normalized it to this standard reference datum. Thus, we have the following expression which yields the pressure, or load, due to wind:

$$\vec{p}_w = 0.00256\vec{V}_{10}^{\,2} \qquad \textbf{Eq. 9}$$

where:

$$\vec{V}_{10} = \left(\frac{h}{32.8084}\right)^{\frac{1}{7}} \vec{V}_w \qquad \textbf{Eq. 10}$$

giving the following relation:

$$\vec{p}_w = 0.00256\left(\left(\frac{h}{32.8084}\right)^{\frac{1}{7}} \vec{V}_w\right)^2 \qquad \textbf{Eq. 11}$$

where:
- p_w = wind pressure, lbf/ft^2,
- h = height above water surface, ft,
- V_w = measured wind speed, mph.

If physical characteristics of the ship are known, the unit wind pressure can be used in the following relation[1] to calculate the total force on the moored ship, and thus, on the pier, due to wind:

$$\bar{P}_w = kp_w C_1 C_2 \left(\sum A_x \sin^2 \theta_w + \sum A_y \cos^2 \theta_w \right) \qquad \textbf{Eq. 12}$$

where:

P_w	=	Total wind force, lbf,
k	=	1.3, shape factor-considers suction increase, leeward side,
p_w	=	wind pressure, lbf/ft^2,
$C1$	=	Coefficient that considers length of ship, see Table 3,
$C2$	=	gust factor; average value range 1.25-1.45, small values for large ships and large values for small ships, see Table 3 for adaptation,
ΣA_x & ΣA_y	=	sum of exposed to wind areas of ship, structure, buildings on the structure, etc. in x and y direction, ft^2,
θ_w	=	angle of wind direction to pier centerline with 0 being straight off the end of the pier from the foot of the pier.

	Length of the Ship			
	≤ 80 ft	165 ft	330 ft	≥ 655 ft
C_1	1	0.80	0.65	0.50
C_2	1.45	1.37	1.31	1.25

Table 3. Coefficients C_1 and C_2 for wind force calculation.

If ship characteristics are unavailable, Tsinker[1] provides a graph relating ship displacement to approximate wind load per unit length of ship. After conversion to the USCS system of units, the plot looks like Figure 9. The values represented in this plot are considered conservative.

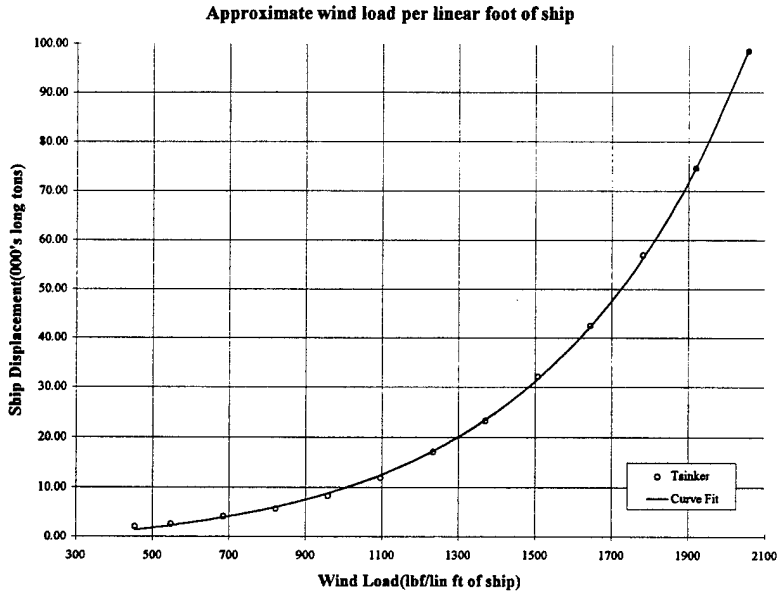


Figure 9. Displacement to wind load relation (Tsinker[1])

However, they do enable the approximation of ship size capacity in the absence of ship dimension data. A curve fit was performed using Origin™ from Microcal™ resulting in the following relation:

$$D_S = -2.97522 + 3.26401e^{\left(\frac{(P'w-302.97055)}{509.74674}\right)} \qquad \textbf{Eq. 13}$$

where: D_S = Ship Displacement, 000's long tons,
P'_w = Wind Load/lin ft of ship, lbf.

If limited information about the ship is known, such as the displacement, the data from Figure 9 can be ordered such that wind loading is a function of ship displacement and the following relation is determined from curve fit:

$$P'_w = 0.13376 + 1.40261\left(1 - e^{\frac{-D_S}{53.09003}}\right) + 0.73301\left(1 - e^{\frac{D_S}{4.30876}}\right) \qquad \textbf{Eq. 14}$$

where P'_w and D_S are as before. If an additional ship were to moor at the pier, the wind load effects resulting form the presence of the second ship would be half of those of the first ship. This is due to sheltering effects of the windward ship on the leeward ship. However, this assessment considers only one ship at the pier.

As stated previously, the maximum wind effects on the pier are when the wind is perpendicular to the pier. If the wind direction is landward along the longitudinal axis of the pier, the deck structure, along with the pier-shore interface, assists the pilings in resisting the load. If the wind direction is seaward, sheltering significantly reduces the wind load since a majority of the exposed ship area is below the land elevation. When designing for wind load, the minimum limiting wind velocity should be taken as 70 miles per hour. This value will be checked in the calculations of wind loading in the program.

3. Current

The current force on the ship and submerged structure can be obtained from the following relation:

$$\vec{P}_c = \frac{1}{2}\rho_w C_D\left(\sum A_x V_{c,x}{}^2 + \sum A_y V_{c,y}{}^2\right) \qquad \textbf{Eq. 15}$$

where:

P_c	=	avg current force submerged object, lbf,
ρ_w	=	density of water, slugs/ft3,
C_D	=	drag coefficient, 0.6-1.2 for piles, 1.0 for ship
A_i	=	exposed area in the *i*th direction, ft^2,
$V_{c,i}$	=	current velocity, ft/s.

giving the average force per unit length of pile over the depth as

$$\vec{P}_{avg} = \frac{\vec{P}_c}{d} \qquad \textbf{Eq. 16}$$

For the pile, the resulting moment at the mudline due to this current can be written as

$$\vec{M}_c = \int_0^d \vec{P}_{avg} z dz = \frac{\vec{P}_{avg} d^2}{2} = \frac{\vec{P}_c d}{2} \qquad \textbf{Eq. 17}$$

The force on the ship due to current will be transferred to the pier as a point load at the pile-cap interface. Generally, the current, if any, flows perpendicular to the pier. The coefficient *c* provides for roughness due to organic growth. For heavy growth, the effective diameter of the piling may need to be increased when determining current forces on the pile.

4. Waves

The following discussion is an adaptation from the U.S. Army Corps of Engineers(COE)[10] on dealing with wave effects on piles. The COE has a comprehensive approach to determining wave forces and moments on piles, often relying on graphs and charts for values of

coefficients or non-dimensionalized factors. This sort of analysis does not work well for programming since it would require limitless effort to curve-fit each and every plot contained therein. However, it is also possible to arrive at these values numerically, given some simplifications.

When assessing the forces on a pile, it is unnecessary to have detailed information about the force distribution along the pile. What is important is the total force acting on the pile and the total moment about the mudline. As discussed earlier, the presence of the pile in the wave field has little, if any, effect on the passing wave due to its size relative to the wavelength. Thus, the Morison equation is applicable and the total wave force, in the direction of wave propagation, is:

$$\vec{P}_{wave} = \vec{p}_{inertial} + \vec{p}_{drag} = C_M \rho \frac{\pi D^2}{4} \frac{d\bar{u}}{dt} + C_D \frac{1}{2} \rho D \bar{u}|\bar{u}| \qquad \textbf{Eq. 18}$$

where:

P_{wave}	=	wave force
p_i	=	component force
C_M, C_D	=	hydrodynamic force coefficients
ρ	=	density of seawater
D	=	diameter of pile
u	=	water particle velocity
du/dt	=	water particle acceleration

The total force, P_{wave}, and total moment of forces, M_{wave}, can be found through integration:

$$P_{wave} = \int_{-d}^{\eta} p_{inertial} dz + \int_{-d}^{\eta} p_{drag} dz = P_{inertial} + P_{drag} \qquad \textbf{Eq. 19}$$

$$M_{wave} = \int_{-d}^{\eta} (z+d) p_{inertial} dz + \int_{-d}^{\eta} (z+d) p_{drag} dz = M_{inertial} + M_{drag} \qquad \textbf{Eq. 20}$$

In general form, these quantities can be written

$$P_{inertial} = C_M \rho g \frac{\pi D^2}{4} H_{wave} K_i \qquad \textbf{Eq. 21}$$

$$P_{drag} = C_D \frac{1}{2} \rho g D H_{wave}^{\ 2} K_D \qquad \textbf{Eq. 22}$$

$$M_{inertial} = C_M \rho g \frac{\pi D^2}{4} H_{wave} K_i S_i d \qquad \textbf{Eq. 23}$$

$$M_{drag} = C_D \frac{1}{2} \rho g D H_{wave}^{\ 2} K_D S_D d \qquad \textbf{Eq. 24}$$

where: g = gravity
H_{wave} = wave height
d = depth

From this point on we must assume that Airy theory applies to be able to complete the analysis of wave forces. A check is required to ensure that the conditions do fall within the Airy regime. If they do not, the analysis will proceed using Airy theory with a note of caution in the results. The literature provides a plot describing the regions of validity for the various wave theories using the wave height and the depth. This plot includes the various Stokes' theories and the Cnoidal theory regions. Reading data off of the plot and performing a curve fit yields the following criteria for validity of Airy wave theory:

$$\frac{d}{gT_{wave}^{\ 2}} \le 0.07 \quad and \quad \frac{H_{wave}}{gT_{wave}^{\ 2}} \le \left[0.00103 - \frac{0.0017}{\left(1 + e^{\frac{\left(\frac{d}{gT_{wave}^{\ 2}} - 0.00549 \right)}{0.01306}} \right)} \right] \qquad \textbf{Eq. 25}$$

$$\frac{d}{gT_{wave}^{\ 2}} > 0.07 \quad and \quad \frac{H_{wave}}{gT_{wave}^{\ 2}} \le 0.00103 \qquad \textbf{Eq. 26}$$

where: T_{wave} = wave period
d = depth
H_{wave} = wave height.

If either Eq. 25 or Eq. 26 are valid, then Airy wave theory applies in which case the following relations apply:

$$K_i = \frac{1}{2}\tanh\left(\frac{2\pi d}{\lambda}\right)\sin\left(\frac{-2\pi t}{T_{wave}}\right) \qquad \textbf{Eq. 27}$$

$$K_D = \frac{1}{8}\left(1 + \frac{4\pi d/\lambda}{\sinh(4\pi d/\lambda)}\right)\left|\cos\left(\frac{2\pi t}{T_{wave}}\right)\right|\cos\left(\frac{2\pi t}{T_{wave}}\right) \qquad \textbf{Eq. 28}$$

$$S_i = 1 + \frac{1 - \cosh(2\pi d/\lambda)}{(2\pi d/\lambda)\sinh(2\pi d/\lambda)} \qquad \textbf{Eq. 29}$$

$$S_D = \frac{1}{2} + \frac{1}{4\left(1 + \frac{4\pi d/\lambda}{\sinh(4\pi d/\lambda)}\right)}\left(\frac{1}{2} + \frac{1-\cosh(4\pi d/\lambda)}{(4\pi d/\lambda)\sinh(4\pi d/\lambda)}\right) \qquad \textbf{Eq. 30}$$

From Eq. 27 and Eq. 28, the maximum values of the various force and moment components can be rewritten and Eq. 21 through Eq. 24 can now be further simplified:

$$\vec{P}_{inertial,m} = C_M \rho g \frac{\pi D^2}{4} H_{wave} K_{i,m} \qquad \textbf{Eq. 21a}$$

$$\vec{P}_{drag,m} = C_D \frac{1}{2} \rho g D H_{wave}{}^2 K_{D,m} \qquad \textbf{Eq. 22a}$$

$$\vec{M}_{inertial,m} = \vec{P}_{inertial,m} S_i d \qquad \textbf{Eq. 23a}$$

$$\vec{M}_{drag,m} = \vec{P}_{drag,m} S_D d \qquad \textbf{Eq. 24a}$$

where $K_{i,m}$ and $K_{D,m}$ according to Airy theory are obtained taking $t = -T_{wave}/4$ and $t = 0$ respectively. Depending upon depth, the inertial term is much smaller than the drag term and can be neglected. However, calculations should be performed across a wave cycle and the maximum combined value used for the assessment. When a wave is moving through the pile group, the maximum loading is not on every pile. However, since at some point it will be, the lateral capacity for each pile will be reduced accordingly. The moments above will be combined with the moment from the lateral load induced by the ship and the lateral load induced by the current to determine the maximum allowable lateral load.

For the purposes of this assessment, the hydrodynamic force coefficients can be approximated as seen in Table 4.

Reynolds Number criteria	Coefficient
$R_e < 3 \times 10^5$	$C_D = 1.2$
$3 \times 10^5 < R_e$	$C_D = 0.6$
$R_e < 2.5 \times 10^5$	$C_M = 2.0$
$2.5 \times 10^5 < R_e < 5 \times 10^5$	$C_M = 2.5 - R_e/(5 \times 10^5)$
$R_e > 5 \times 10^5$	$C_M = 1.5$

Table 4. Hydrodynamic coefficients, C_D and C_M

The Reynolds number is given by

$$R_e = \frac{u_{\max} D}{\nu} \qquad \textbf{Eq. 31}$$

where: u_{max} = maximum particle velocity
D = pile diameter
ν = 1.0×10^{-5} ft^2/s, kinematic viscosity seawater

and u_{max} is given by

$$u_{\max} = \frac{\pi H_{wave} \lambda_0}{T_w \lambda_A} \qquad \textbf{Eq. 32}$$

λ_0 is the deep water wavelength

$$\lambda_0 = \frac{g T_{wave}^2}{2\pi} \qquad \textbf{Eq. 33}$$

and λ_A is the local wavelength

$$\lambda_A = \frac{g T_{wave}^2}{2\pi} \tanh\left(\frac{2\pi d}{\lambda_A}\right) \qquad \textbf{Eq. 34}$$

where: d = depth

Since λ_A appears on both sides of Eq. 34, it is necessary to solve using a root finding algorithm such as Newton-Raphson or by using the following approximation provide by Professor D.K.P. Yue, Department of Ocean Engineering, Massachusetts Institute of Technology. We need to first determine the wave regime, deeper water or shallower water. Let

$$c = \frac{\omega^2 d}{g} = \frac{4\pi^2 d}{T_{wave}^2 g} \qquad \textbf{Eq. 35}$$

we can then use the value of c to determine which approximation to use to find the wave number, k. If

$$c > 2 \rightarrow k \approx \frac{c}{d}\left(1 + 2e^{-c} + \cdots\right) \qquad \textbf{Eq. 36}$$

else, if

$$c < 2 \rightarrow k \approx \frac{\sqrt{c}}{d}(1 + 0.169c + \cdots) \qquad \textbf{Eq. 37}$$

thus giving a value for λ_A from the relation for the wave number k

$$k = \frac{2\pi}{\lambda_A} \Rightarrow \lambda_A = \frac{2\pi}{k} \qquad \textbf{Eq. 38}$$

When considering the incident waves upon the moored ship, the lateral force is a function of the reflection coefficient, wave height, depth, and period. For the depth in which a ship will be tied up alongside the pier, the waves will be non-breaking during normal weather. Thus, the force is due to the increase in hydrostatic head caused by the wave. In extreme circumstances, the increase in pressure per linear foot of ship will be

$$\Delta P = \frac{1}{2}\rho_w g H_{wave} \qquad \textbf{Eq. 39}$$

but will more likely be

$$\Delta P < \frac{1}{2}\rho_w g \frac{H_{wave}}{3} \qquad \textbf{Eq. 40}$$

and will be neglected since the wave height required to generate any significant force on the ship will have to be ≈ 10% of the depth or greater which in a depth of 30-35' is 3-3.5' waves. If the seas are this rough, other considerations will prevail for the safe mooring of the vessel.

5. Dynamic

Regarding dynamic analysis, Gaythwaite[3] states that "Most waterfront structures can be analyzed using static methods..." As stated previously, marine structures, such as piers, are designed for high lateral loading and thus, are relatively rigid with natural periods on the order of 0.5 seconds or less[3]. Additionally, immersion in water provides damping in addition to that inherent in land-based structures. This study covers only those structures immediately adjacent to the shore which may be required for operation in a tactical situation. However, there may be exceptions such as piers having vertical cantilever piles. Of the two predominant contributions to dynamic effects, periodic waves are of the most concern for everyday operations. If a wave having a period close to that of the moored ship is incident to the ship, it may result in resonant phenomena that will amplify the ships mooring forces on the pier. Without an in-depth, on-site analysis, the best that can be done is to approximate the period of the pier and compare it with that of the incident waves. Our primary concern is those piers that have no cross bracing and no batter piles. Piers of this type can be idealized as a single degree

of freedom system with the pilings modeled as cantilever springs and the natural frequency estimated using the following:

$$\omega_{n,pier} = \sqrt{\frac{k_{eq}}{m}} \qquad \textbf{Eq. 41}$$

where: $w_{n,pier}$ = natural frequency of the pier, rad/s,
m = mass of the pier + a portion of the live load,
k_{eq} = equivalent spring constant

with k_{eq} given by:

$$k_{eq} = k + k_2 + k_3 + \cdots + k_n \qquad \textbf{Eq. 42}$$

where, for a cantilever spring, k_n, is given by:

$$k_n = \frac{3EI}{L_{eff}^3} \qquad \textbf{Eq. 43}$$

with: E = modulus of elasticity for the piling material, psi,
I = moment of inertia of the pile, in^4,
L_{eff} = effective length of the pile, in.

and for a round pile,

$$I = \frac{\pi r^4}{4} \qquad \textbf{Eq. 44}$$

where: r = radius of pile, in.

If the piling has experience damage and has a reduced cross-section, it will be assumed to be a rectangular beam whose shape will fit in the remaining pile cross-section. Thus:

$$I = \frac{bh^3}{12} \qquad \textbf{Eq. 45}$$

where: b = base of the rectangle, in,
h = height of the rectangle, in (taken as the smaller dimension in the case of pile damage).

The period is then found from:

$$T_{pier} = \frac{2\pi}{\omega_{n,pier}} \qquad \textbf{Eq. 46}$$

Should there be any structures or installed equipment on the pier, this will increase the mass and decrease the natural frequency.

B. Piles

Pilings are the primary support for the waterfront structure. The are also the structural members most likely to experience damage. During vertical loading they act like a column. During lateral loading they can act like a cantilever beam. Thus, one of the most critical aspects to a pile is its length. The Naval Facilities Engineering Service Center, formerly Naval Civil Engineering Lab, recommends reducing the published ultimate strengths for fir and pine pilings that have been treated. However, the values used in this assessment are published allowable values and include a factor of safety which should adequately allow for the reduction in strength due to treatment while maintaining a conservative capacity estimate.

1. Fixity

Fixity is that point in the soil from which the unsupported length is calculated. Typically, soil strength information would be available to calculate this using developed relations. However, since this information is unavailable to UCT personnel, some simplifying assumptions are required. Gaythwaite[3] states the depth to fixity, D_f, usually lies within a range of 3.5 to 8.5 pile diameters. NAVFAC[6] provides more detailed guidance in the absence of the coefficient of sub-grade reaction (EI is the modulus of elasticity to moment of inertia ratio of the pile.):

Soft, cohesive soils $EI \leq 10x10^9$ psi $D_f = 10$ ft

$EI > 10x10^9$ psi $D_f = 12$ ft

Loose, granular soils & medium, cohesive soils

$EI \leq 10x10^9$ psi $D_f = 8$ ft

$EI > 10x10^9$ psi $D_f = 10$ ft

For other cases $D_f = 5$ ft.

The UCT's possess the capability to perform a Rapid Penetration Test, which is a limited near-shore geotechnical survey. The only results from that test that are useful to this assessment are those indicating the type of soil present at the site. The type of soils determined from the RPT can be correlated directly with the above NAVFAC guidance regarding D_f. For the purposes of this assessment, the RPT results will compare as shown in Table 5. If RPT test results are not available, D_f will be assumed to be 8.5 times the pile diameter. Figure 10 illustrates the concept of depth to fixity. In the case of a batter pile, D_f remains the same.

However, the resulting unsupported length is longer than that in the case of the vertical pile since the additional unsupported length for the batter is equal to $D_f/\sin\theta$ where θ is the angle the pile makes with the vertical.

NAVFAC	RPT
Soft, cohesive soils	Very soft clay
	Soft clay
	Soft silt
	Mud
Loose, granular soils & medium, cohesive soils	Medium clay
	Loose sand
Other cases	Medium sand
	Stiff clay
	Dense sand & gravel

Table 5. Soil compatability table for D_f.

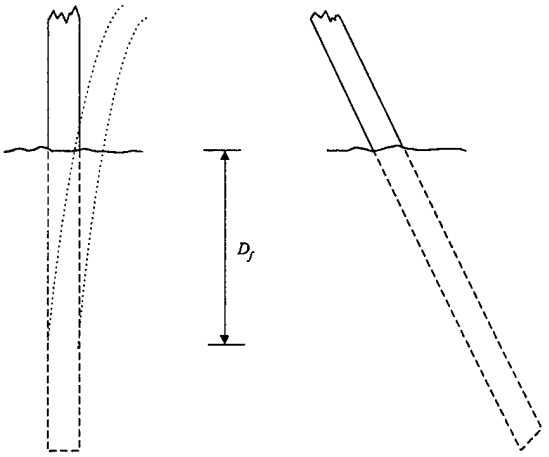


Figure 10. Depth to fixity illustrated.

2. Vertical

The pilings supporting the pile behave as columns. The embedded end of the piling is considered fixed for the purposes of this assessment. The pile cap end, however, will be configured either as pinned or as fixed. The prime difference between the two configurations is the presence of cross-bracing in the latter case which restrains the upper end from rotating. To determine the unsupported length L_u, the distance from the mudline to either the cross-bracing, if so configured, or the pile cap is added to D_f. L_{eff} is dependent upon the end

conditions of the pile. For the two configurations looked at in this assessment, Figure 11 illustrates the end conditions and the corresponding effective lengths, L_{eff}.

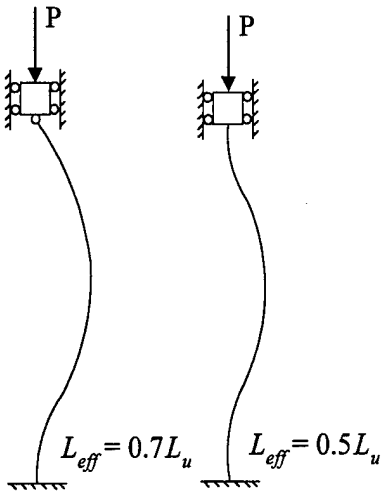


Figure 11. Bending modes and effective lengths of 2 columns.

Where this makes a difference as far as this assessment is concerned is in computing the working stress of the piling. Since the pilings behave as columns, *Euler's formula* for critical buckling is applicable. This relation is as follows:

$$P_{cr} = \frac{\pi^2 EI}{L_{eff}^{\ 2}} \qquad \textbf{Eq. 47}$$

where:
- P_{cr} = critical load, lbf,
- E = modulus of elasticity, psi,
- I = moment of inertia, in^4,
- L_{eff} = effective length, in.

This leads to the critical stress:

$$\sigma_{cr} = \frac{P_{cr}}{A} \qquad \textbf{Eq. 48}$$

where:
- σ_{cr} = critical stress, psi,
- A = area of cross section, in^2.

The allowable stress is the maximum, or critical, stress reduced by the factor of safety. Once this is accomplished, the allowable loading can be determined as follows:

$$\sigma_{all} = \frac{\sigma_{cr}}{FOS} \qquad \textbf{Eq. 49}$$

where:
- σ_{all} = allowable stress, psi,
- σ_{cr} = critical stress, psi,

FOS = factor of safety.

However, this value must be compared to published values for allowable stresses before proceeding with load calculations. The lesser of the two values, henceforth designated σ_{all}, is used in determining allowable axial loading for bearing piles:

$$P_{pile,vert} = \frac{\pi D^2}{4}\sigma_{all} \quad circular \qquad \textbf{Eq. 50}$$

$$= bh\sigma_{all} \quad rectangular\ cross-sections \qquad \textbf{Eq. 50a}$$

Tsinker discusses a factor of safety for pile loading as follows:

> *"The working load on a pile is defined as the ratio of pile ultimate capacity to the appropriate factor of safety. Generally, the safety factor for a single pile, 2.5 is considered as appropriate. However, where there is a sufficient number of pile loading tests, or where a large body of load experience is available, the safety factor of 2.0 is normally considered. In cases where the soil characteristics are uncertain or large impact or vibratory loads are expected, the safety factor of up to 3.0 and more can be considered. For temporary structures, depending on site geological condition and pile loading, the factor of safety can be reduced to 1.5-2.0."*[1]

Vertical loading on pilings will be discussed further when developing the assessment procedures for pile caps.

3. Lateral

The lateral resistance of the structure will be provided by the bearing piles and the batter piles. If present, the batter pile capacity in lateral loading far exceeds that of the bearing piles. However, both configurations may be encountered and thus are discussed. For bearing pile capacity, we look at the maximum moment the pile can be subjected to given the allowable stresses for bending as published. If the pile is not braced and merely pinned into the cap, it is considered a pinned-fixed column. However, it experiences loading as a cantilever beam. The maximum moment for a cantilever beam is expressed as:

$$M = PL_{eff} \qquad \textbf{Eq. 51}$$

In the case of pilings that are braced, they are considered fixed-fixed and the maximum moment experienced due to a load is

$$M = \frac{PL_{eff}}{2} \qquad \textbf{Eq. 52}$$

Given the stress due to bending[11]

$$\sigma = \frac{M}{S} \qquad \textbf{Eq. 53}$$

where

$$S = \frac{I}{c},\ elastic\ section\ modulus \qquad \textbf{Eq. 54}$$

$$= \frac{\pi r^3}{4}\ for\ a\ circular\ pile \qquad \textbf{Eq. 54a}$$

$$= \frac{bh^2}{6}\ for\ a\ rectangular\ cross-section \qquad \textbf{Eq. 54b}$$

we arrive at a relation between the allowable stress, σ_{all}, and the induced moments

$$\sigma_{all,bend} S = M_{bending,ship} + M_{wave} + M_{current} + M_{wind} \qquad \textbf{Eq. 55}$$

which then gives us the allowable lateral loading

$$\vec{P}_{pile,lat} = \frac{2}{L_{eff}}\left(\sigma_{all}\frac{\pi r^3}{4} - \bar{M}_{wave} - \bar{M}_{pile,current}\right)\ for\ a\ circular\ pile - braced \qquad \textbf{Eq. 56a}$$

$$= \frac{1}{L_{eff}}\left(\sigma_{all}\frac{\pi r^3}{4} - \bar{M}_{wave} - \bar{M}_{pile,current}\right)\ for\ a\ circular\ pile - pinned \qquad \textbf{Eq. 56b}$$

$$= \frac{1}{L_{eff}}\left(\sigma_{all}\frac{bh^2}{6} - \bar{M}_{wave} - \bar{M}_{pile,current}\right)\ for\ a\ rectangular\ cross-section \qquad \textbf{Eq. 56c}$$

Finally, adding in Eq. 17, Eq. 23, and Eq. 24, we have

$$\vec{P}_{pile,lat} = \frac{2}{L_{eff}}\left(\sigma_{all}\frac{\pi r^3}{4} - \left(C_M\rho g\frac{\pi D^2}{4}H_{wave}K_i S_i d + \frac{1}{2}C_D\rho g D H_{wave}{}^2 K_D S_D d\right)_{max} - \frac{d}{4}\rho_w C_D A\vec{V}_c^{\,2}\right) \qquad \textbf{Eq. 57a}$$

$$= \frac{1}{L_{eff}}\left(\sigma_{all}\frac{\pi r^3}{4} - \left(C_M\rho g\frac{\pi D^2}{4}H_{wave}K_i S_i d + \frac{1}{2}C_D\rho g D H_{wave}{}^2 K_D S_D d\right)_{max} - \frac{d}{4}\rho_w C_D A\vec{V}_c^{\,2}\right) \qquad \textbf{Eq. 57b}$$

$$= \frac{1}{L_{eff}}\left(\sigma_{all}\frac{bh^2}{6} - \left(C_M\rho g\frac{\pi D^2}{4}H_{wave}K_i S_i d + \frac{1}{2}C_D\rho g D H_{wave}{}^2 K_D S_D d\right)_{max} - \frac{d}{4}\rho_w C_D A\vec{V}_c^{\,2}\right) \qquad \textbf{Eq. 57c}$$

For batter piles, we refer back to Euler's formula (Eq. 47) and the published allowable stress in compression parallel to the grain to find the maximum allowable load on a batter.

$$P_{pile,batter,lat} = \frac{\pi D^2}{4} \sigma_{all,comp} \cos\theta_{batter} \quad \textbf{Eq. 58}$$

Typically, a batter pile can resist lateral load in tension as well as in compression. However, detailed knowledge of the soil characteristics are required to conducted a thorough analysis of this capacity. Tsinker[1] states that, based upon additional geotechnical information, it is customary to assume 2/3. For the purposes of this assessment, the value will be assumed to be 1/3 that of the axial capacity[7].

C. Decking

The decking, which includes the structural members from the pile cap up, can all be modeled as beams. The stringers are modeled as simply supported beams while the pile caps and the planking are modeled as continuous beams.

1. Stringers - Simply Supported Beams

A simply supported beam is supported at each end and has a single span. It is statically determinate and develops a maximum moment for a concentrated load when the load is placed at the midpoint. The value of this moment is equal to

$$M = \frac{P_{conc} L}{4} \quad \textbf{Eq. 59}$$

If a uniform load is present across the entire span, it creates the following moment

$$M = \frac{P_{uni} L^2}{8} \quad \textbf{Eq. 60}$$

For distribution of wheel loads to stringers, AASHTO[7] provides an adjustment to the live load bending moment for timber plank floors as follows:

$$Adj_{mom} = \frac{S}{4}$$

where: S = span of stringers, ft

The allowable load can now be found in a manner similar to that for the pile in bending using Eq. 53

$$P_{st,all} = P_{conc,all} = \frac{4\sigma_{all} S}{(bent\ spacing) Adj_{mom}} = \frac{2 b_{stringer} h_{stringer}^{\ 2}}{3(bent\ spacing) Adj_{mom}} \sigma_{all,bend} (lb) \quad \textbf{Eq. 61a}$$

$$= P_{uni,all} = \frac{8\sigma_{all} S}{Adj_{mom} (bent\ spacing)^2} = \frac{4 b_{stringer} h_{stringer}^{\ 2}}{3 Adj_{mom} (bent\ spacing)^2} \sigma_{all,bend} (lb / in) \quad \textbf{Eq. 61b}$$

where:

$b_{stringer}$	=	stringer base dimension, in
$h_{stringer}$	=	stringer height dimension, in
$\sigma_{all,bend}$	=	allowable stress, psi
bent spacing	=	spacing of bents, in

The uniform loading can be converted to a *psf* capacity if we consider tributary loading analysis, that is, the stringer carries half the planking span load on either side. Thus, from Eq. 61b the stringer can carry a uniform load of:

$$P_{uni,all,tot} = \frac{P_{uni,all}}{stringer\ spacing} \quad \textbf{Eq. 62}$$

Since a pier will be subjected to various types of point loading such as cargo handling equipment and vehicular traffic AASHTO[7] specifications apply. Figure 12 illustrates standard types and configurations of truck loading contained therein. The designations include either H for a two axle truck or HS for a tractor truck and the gross tons of the truck and tractor truck (w/ 1st axle) respectively.

a. HS Truck Loading

The variable spacing in Figure 12 for the HS trucks account for the various tractor trucks available on the road today. When calculating the moment, the variable spacing, V, is to be adjusted from 14-30ft to provide the maximum value of moment. During cargo handling operations, trucks will be transiting the length of the pier in both directions. However, fully loaded trucks should be transiting in only one direction. Thus, load capacity specifications should be similar to those for a single lane bridge as presented by AASHTO[7].

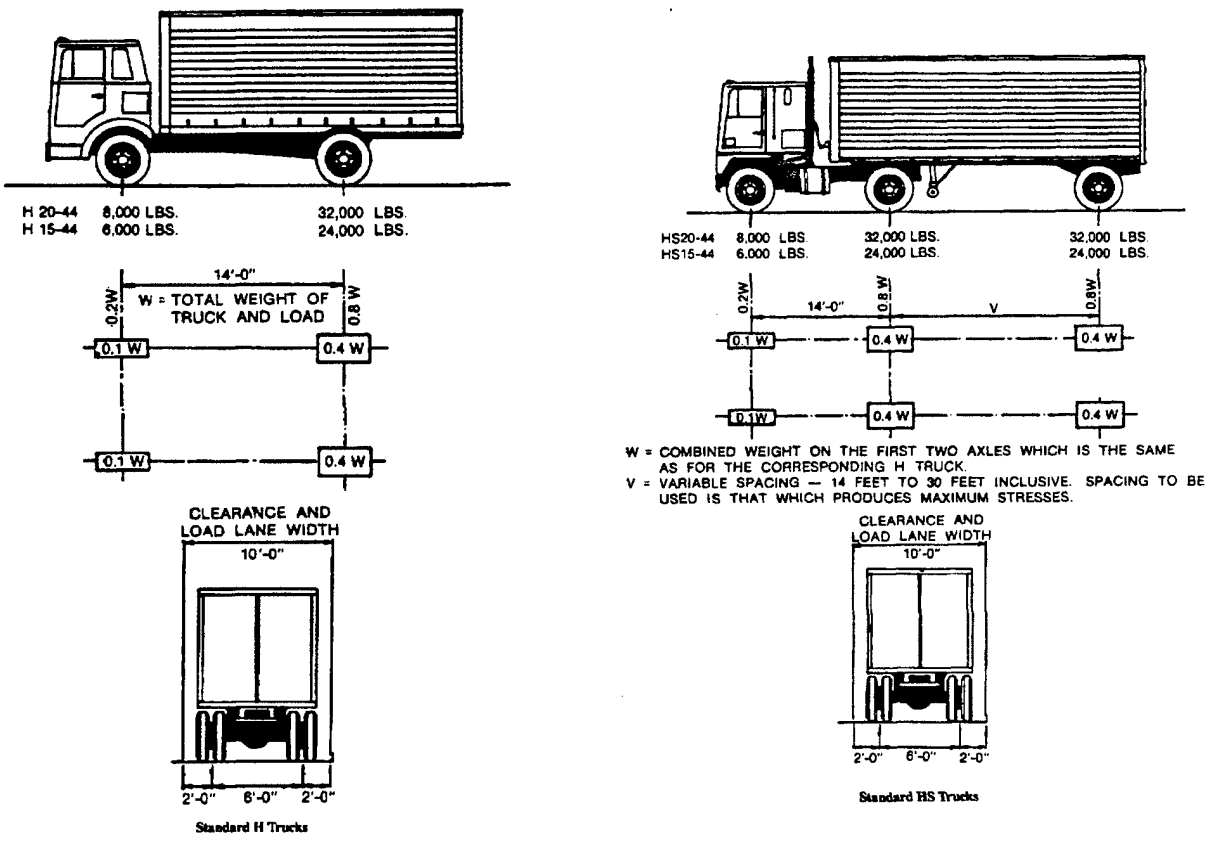


Figure 12. AASHTO Truck specifications

Since the truck will be traveling along the stringers, it is considered a rolling load and thus, the maximum moment calculation is more complex. However, *Marks' Standard Handbook for Mechanical Engineers*[12] provides a solution strategy for this type of loading. For the HS trucks with two evenly loaded axles, on a simply supported beam such as stringers, the following relation applies:

$$M_{\max} = \frac{P}{2L}\left(L - \frac{a}{2}\right)^2 \qquad \textbf{Eq. 63}$$

where: P = the wheel load, lbs,
L = bent spacing, ft,
a = the spacing between axles.

There are some practical limitations to this relation. For example if $L \leq a$, then the maximum moment could be caused by only one wheel load, placed mid-span. Thus, for a range of values of L/a, Eq. 59 applies. To determine which equation should be used to give the maximum moment, we set Eq. 59 equal to Eq. 63

$$M = \frac{P_{conc} L}{4} = \frac{P}{2L}\left(L - \frac{a}{2}\right)^2 = M_{\max}$$

we then get an expression relating L and a

$$\frac{L^2}{2} - La + \frac{a^2}{4} = 0 \qquad \textbf{Eq. 64}$$

Solving Eq. 64 for L, we have

$$L = \begin{bmatrix} a + \frac{a}{2}\sqrt{2} \\ a - \frac{a}{2}\sqrt{2} \end{bmatrix} \qquad \textbf{Eq. 65a} \qquad \textbf{Eq. 65b}$$

Of the two solutions, only Eq. 65a applies since Eq. 65b results in a stringer span shorter than the axle span. Thus, at a ratio of $L/a = 1 + \sqrt{2}/2 = 1.707$, the applicable moment equation changes. To determine the applicable regimes for each of the equations, a plot illustrates the resulting values of each versus the ratio L/a is in Figure 13. From this plot, it is clear that for $L/a \leq 1.707$, Eq. 59 dominates and for $L/a > 1.707$, Eq. 63 is applicable.

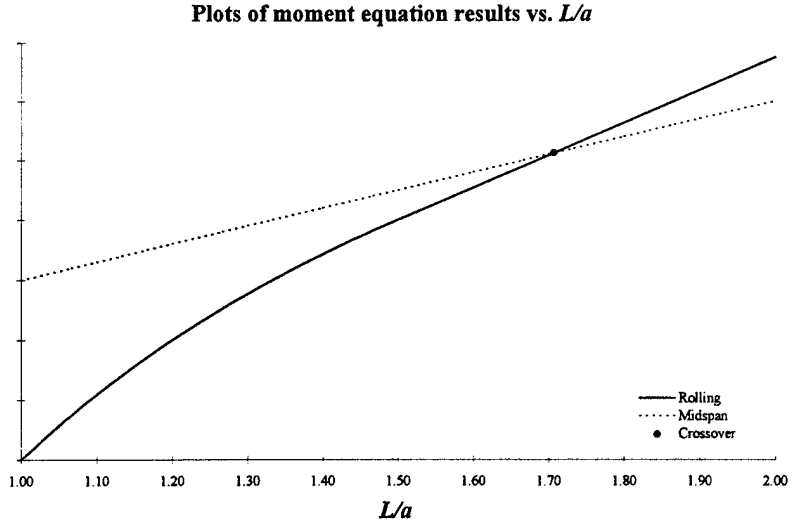


Figure 13. Comparison of moment equation results vs. *L/a* - HS Loading

AASHTO specifies that a should be varied from 14-30' to determine the maximum moment. However, observation of Figure 13 reveals that the moment increases with increasing L/a. This indicates that the smallest value of a, 14', should be used when calculating max moment. Using Eq. 53 and Eq. 54, we now have

$$\frac{bent\ spacing}{a} \le 1.707 \qquad P_{st,all,HS} = \frac{2b_{stringer}h_{stringer}^{\ 2}}{3Adj_{mom}(bent\ spacing)}\sigma_{all,bend}(lb) \qquad \textbf{Eq. 66}$$

$$\frac{bent\ spacing}{a} > 1.707 \qquad P_{st,all,HS} = \frac{(bent\ spacing)b_{stringer}h_{stringer}^{\ 2}}{3Adj_{mom}\left(bent\ spacing - \frac{a}{2}\right)^2}\sigma_{all,bend}(lb) \qquad \textbf{Eq. 66a}$$

Letting $a = 14$, the allowable stringer loading is

$$bent\ spacing \le 23.9 \qquad P_{st,all,HS} = \frac{2b_{stringer}h_{stringer}^{\ 2}}{3Adj_{mom}(bent\ spacing)}\sigma_{all,bend}(lb) \qquad \textbf{Eq. 67}$$

$$bent\ spacing > 23.9 \qquad P_{st,all,HS} = \frac{(bent\ spacing)b_{stringer}h_{stringer}^{\ 2}}{3Adj_{mom}(bent\ spacing - 7)^2}\sigma_{all,bend}(lb) \qquad \textbf{Eq. 68}$$

b. H Truck Loading

H truck loading differs from HS in that it has loads of different magnitudes on the two axles. For H trucks, the axle span is fixed at 14', thus, for any stringer length less than 14', the maximum moment will occur when the heavy axle is mid-span on the beam and Eq. 61a applies. As with the equally loaded axles of the HS trucks, once the value of L exceeds a, or 14' in this case, the combination of point loads on the span has the potential to produce a moment which exceeds that calculated using Eq. 61a. Using Microsoft Excel®, a model was developed to calculate the reaction forces and, using Excel® add-in *Solver*, would iterate the moment arm value to find the maximum moment generated in a simply supported beam. Data was compiled for various values of *L/14* and are displayed in Figure 14. Based upon these results, the maximum moment in the beam for $L / a \ge\approx 1.93$ is a straight line of the form:

$$y = mx + b$$

thus, after a line fit, we have

$$M = \left(4.34475\frac{L}{a} - 1.6009\right)P$$

with $a = 14$,

$$M = (0.3103L - 1.6009)P \qquad \textbf{Eq. 69}$$

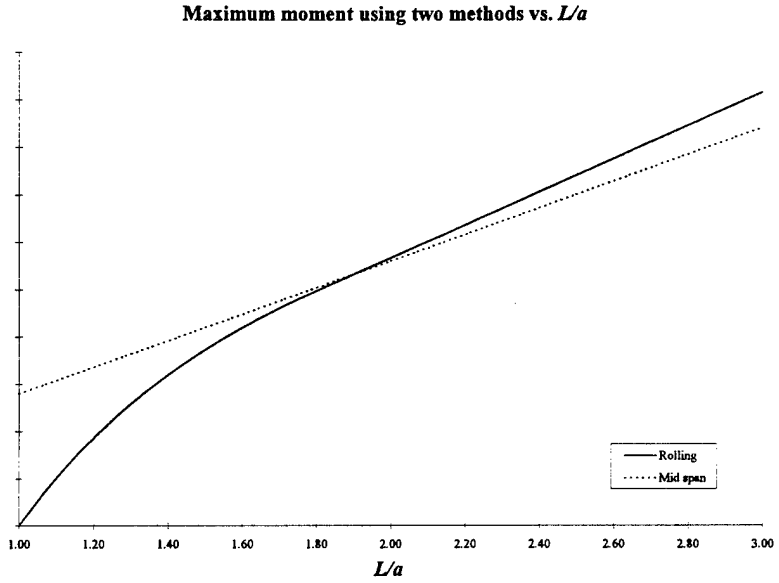


Figure 14. H Loading resulting moments vs. *L/a*

Finally, for the maximum allowable load due to an H truck, we have

bent spacing ≤ 27

$$P_{st,all,H} = \frac{2b_{stringer}h_{stringer}^{\ 2}}{3Adj_{mom}\left(bent\ spacing\right)}\sigma_{all,bend}\left(lb\right) \qquad \textbf{Eq. 70}$$

bent spacing > 27

$$P_{st,all,H} = \frac{b_{stringer}h_{stringer}^{\ 2}}{Adj_{mom}\left(1.862\left(bent\ spacing\right)-9.606\right)}\sigma_{all,bend}\left(lb\right) \qquad \textbf{Eq. 71}$$

c. Forklift Loading

Table 6 is taken from NAVFAC[4] and provides the maximum wheel loads for a number of

			Wheel Loads (Loaded)	
Max Load (lbs)	Wheel Base (ft-in)	Wheel Space (ft-in)	Each Rear Single (lbs)	Each Front Dual (lbs)
10,000	8-3	6-3	2,000	10,000
12,000	8-3	6-3	2,500	11,500
15,000	8-9	6-4	2,500	14,500
16,000	8-9	6-4	2,500	15,250
20,000	9-6	6-4	2,500	17,500
24,000	10-0	6-4	2,500	22,150
30,000	10-9	6-6	3,000	29,000
40,000	10-0	8-0	2,500	49,000

Table 6. Forklift wheel loads and dimensions.

different sized forklifts. As can be seen, the minimum spacing of wheels occurs with the front wheels which are also equally, and the heaviest, loaded. Therefore, when checking for forklift capacity on the pier, Eq. 66 and Eq. 66a can be used with the appropriate value of *Wheel spacing* inserted for *a*.

d. Crane Loading

Appendix D contains the load data[4] on a number of mobile cranes. The outrigger loads therein can be considered as either point loads or uniform loads if reduced by the outrigger area. The program will provide maximum loading capacity for stringers and pile caps. It is left up to the on-site personnel to enter the tables and determine the maximum crane loading.

e. Stringers - A Final Note

All of the analysis regarding stringers has been conducted under the assumption that, as the loaded truck transits the pier, a stringer will have to bear a single wheel load from each axle. This assumption is fairly safe given that the wheel spacing on both trucks and forklifts is 6' or larger and the span on point loads for cranes is larger still. Should the stringer spacing exceed 6', the above analysis may be less accurate. However, for timber piers, the stringer spacing is usually well within the range considered. Additionally, as the stringer spacing increases, the *planking* load carrying ability will begin to dominate before the above analysis begins to break down.

2. Continuous Beams

Continuous beams "...[have] two or more spans (i.e., three or more supports) and is statically indeterminate."[13] The EIT Reference manual provides a method for determining the reactions on continuous beams using the *three-moment equation*. When analyzing a beam with more than two spans, the equation must be used with three adjacent supports at a time, starting with a support whose moment is known.[13] For a pile cap, this is the end where the moment is zero. In the most general form, the three moment equation can be used with beams of varying cross-section and looks like

$$\frac{M_k L_k}{I_k} + 2M_{k+1}\left[\frac{L_k}{I_k} + \frac{L_{k+1}}{I_{k+1}}\right] + \frac{M_{k+2}L_{k+1}}{I_{k+1}} = -6\left[\frac{A_k a}{I_k L_k} + \frac{A_{k+1} b}{I_{k+1} L_{k+1}}\right] \qquad \textbf{Eq. 72}$$

However, for beams of constant cross-section, such as pile caps and planking, the moment of inertia terms can be eliminated leaving

$$M_k L_k + 2M_{k+1}\left[L_k + L_{k+1}\right] + M_{k+2} L_{k+1} = -6\left[\frac{A_k a}{L_k} + \frac{A_{k+1} b}{L_{k+1}}\right] \qquad \textbf{Eq. 73}$$

Figure 15 illustrates the terms present in Eq. 73. To analyze the continuous beam configuration, a Microsoft Excel® model was developed that include 8 supports, 7 spans. This model resulted in a system of *k-1* equations with *k-1* unknowns. Unit loads representing vehicle axle loads were "placed" on the beam and the equations were solved using the Excel® add-in *Solver*. In this way, the percentage of the axle load carried by a particular support could be calculated. The maximum moment is calculated as well. A sample results screen can be seen in Figure 16.

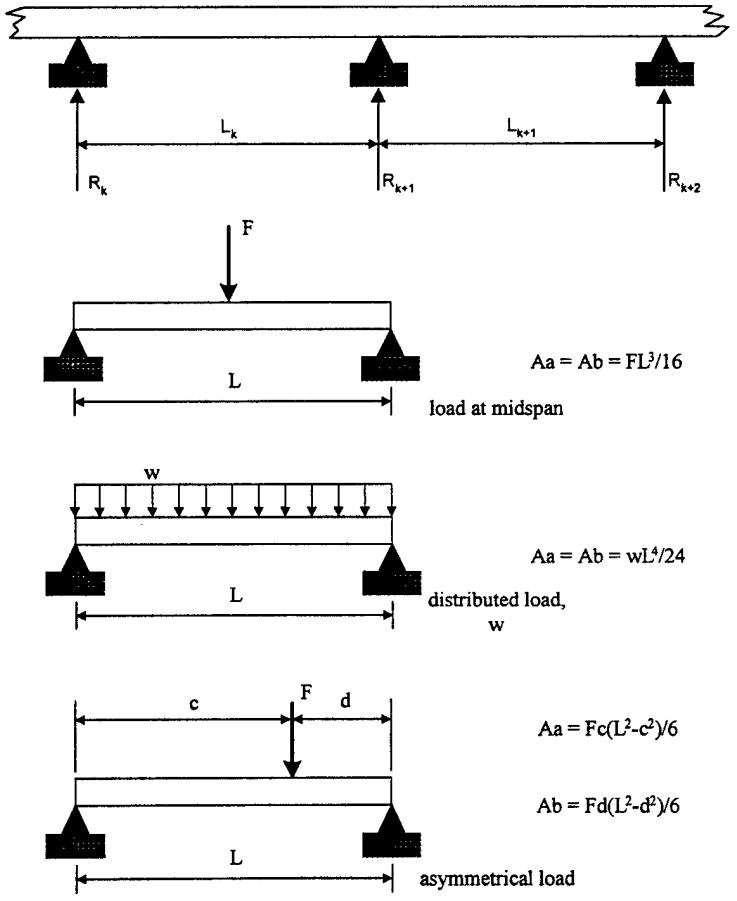


Figure 15. Simplified Three-Moment Equation Terms[13]

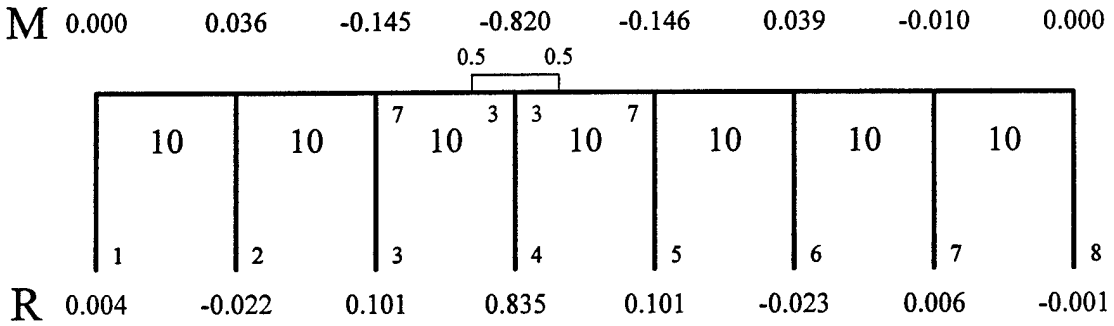


Figure 16. Three-moment equation model output

a. Pile Caps

The three-moment equation model was used to evaluate the impact of pile spacing on the loading of the pile cap. Figure 17 and Figure 18 illustrate the effects of increasing the pile spacing from 6' to 10'. As can be seen, and as expected, the moment and shear increase.

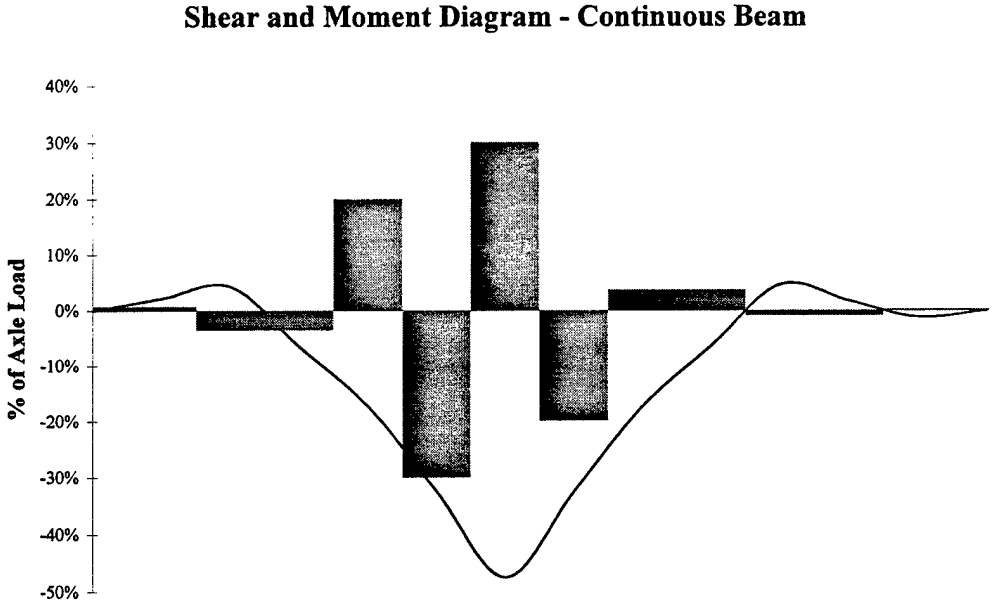


Figure 17. Shear-Moment diagram: Pile spacing - 6'; Axle - centered

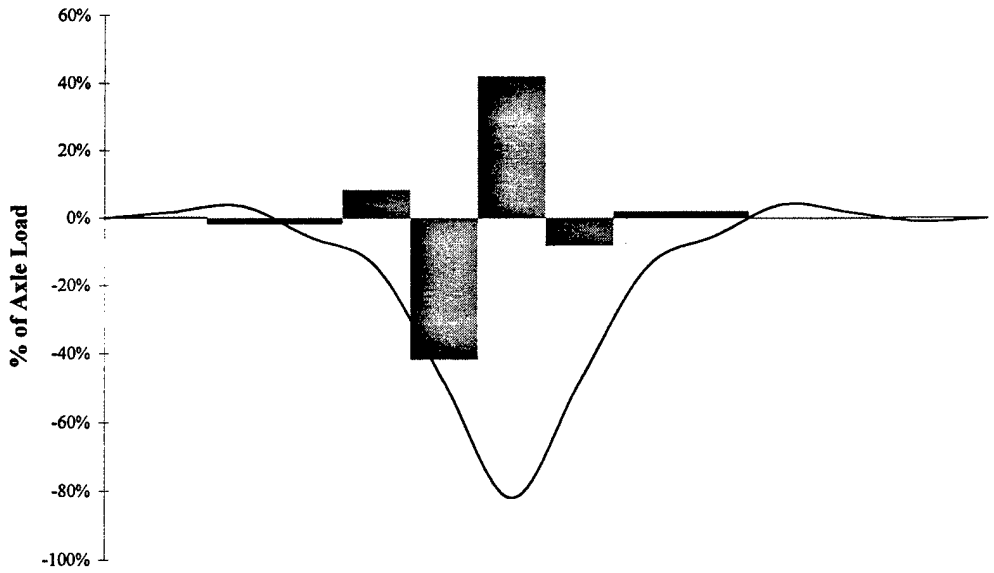


Figure 18. Shear-Moment diagram: Pile spacing - 10'; Axle - centered

Data was collected on the maximum values as the axle position was varied over the support and across the beam. For example, the loads were placed in such a way as to model one wheel load resting directly over a support while the other extends 6' into the span. (6' is the standard axle width considered in design) The calculations were run and the resulting maximum moments and support loadings recorded. This operation was performed for a number of variations of pile spacing and axle placement. A curve was then fit to the collected data providing the maximum moment in the pile cap as a function of the pile spacing and the axle load applied

$$M_{\max} = \left[-124.02 + 92.39\left(1 - e^{\frac{-\text{pile spacing}}{13.59}}\right) + 169.237\left(1 - e^{\frac{-\text{pile spacing}}{3.203}}\right) \right] P \qquad \textbf{Eq. 74}$$

Combining Eq. 74 with Eq. 53, we obtain the maximum allowable axle loading for the pile cap. The maximum wheel load would be half of this value.

$$P_{all,cap} = \frac{\sigma_{all,bend} b_{pile\,cap} h_{pile\,cap}^{\,2}}{6\left[-124.02 + 92.39\left(1 - e^{\frac{-\text{pile spacing}}{13.59}}\right) + 169.237\left(1 - e^{\frac{-\text{pile spacing}}{3.203}}\right) \right]} \qquad \textbf{Eq. 75}$$

The data regarding maximum support loading indicates what percentage of an axle load is transferred to the individual piles. When a curve is fit to the data, and the resulting expression combined with that of allowable loading, $P_{all} = A\sigma_{all}$, the maximum allowable axle load on the pile cap as limited by the pile is given as follows

$$P_{pile\,cap,all,pile} = \frac{A_{pile\,cross\,section}\sigma_{all,comp,pile}}{\left[-134.02 + 213.63\left(1 - e^{\frac{-pile\,spacing}{2.698}}\right) + 22.363\left(1 - e^{\frac{-pile\,spacing}{13.116}}\right)\right]} - dead\,load \qquad \textbf{Eq. 76}$$

where the numerator is the applicable expression from Eq. 50, Eq. 50a. It is unlikely that the pile will be the limiting factor on pile cap loading, but in the case of a damaged pile, Eq. 76 may apply.

When the model results for maximum moment due to uniform loading are evaluated and fit to a curve, we have

$$M = 0.1056337LW \qquad \textbf{Eq. 77}$$

which when combined with Eq. 53 and the tributary area of a pile cap, which is equal to the bent spacing, gives us the maximum allowable uniform loading on the pile caps

$$W_{all} = \frac{b_{pile\,cap}h_{pile\,cap}^{\ 2}}{6 \cdot 0.1056338(pile\,spacing)(bent\,spacing)}\sigma_{all,bend} \qquad \textbf{Eq. 78}$$

If a pile is severely damaged or missing, the pile cap will span twice the pile spacing making it susceptible to increased moments. This situation can be modeled using the three-moment equation with the assumption that only a single pile will be missing from a bent. The model was evaluated with the missing pile first being placed in the middle of the bent. It was then moved to a position next to an exterior pile. The maximum moments transferred to the pile cap occurred in the latter case and it is this equation that will be used to determine the maximum point load on a pile cap with a pile missing.

$$P_{all} = \frac{b_{pile\,cap}h_{pile\,cap}^{\ 2}}{6 \cdot 0.26169(pile\,spacing)}\sigma_{all,bend} \qquad \textbf{Eq. 79}$$

The resulting maximum uniform loading was evaluated in the same fashion with the following result

$$W_{all} = \frac{b_{pile\,cap} h_{pile\,cap}^{\,2}}{6 \cdot 0.1996 (pile\,spacing)(bent\,spacing)} \sigma_{all,bend} \qquad \textbf{Eq. 80}$$

The previous discussion focused on an interior pile being severely damaged or missing. If this is the case with an exterior pile, the pile cap and loading is treated as a simple cantilever beam with either point loading or uniform loading which creates a moment. The following relations apply.

$$P_{all} = \frac{b_{pile\,cap} h_{pile\,cap}^{\,2}}{6(pile\,spacing)} \sigma_{all,bend} \qquad \textbf{Eq. 81}$$

$$W_{all} = \frac{2 b_{pile\,cap} h_{pile\,cap}^{\,2}}{6(pile\,spacing)^2 (bent\,spacing)} \sigma_{all,bend} \qquad \textbf{Eq. 82}$$

b. Planking

The deck planking is also modeled as a continuous beam. As such, the three moment equation model is useful in determining allowable loading. However, for the planking, as with the stringers, wheel loads were analyzed instead of axle loadings. This is because of the relative sizes of the wheel, the wheel loading area, and the stringer spacing, which is the planking span. Additionally, since the plank dimensions and the wheel dimensions are of the same order, the loading is better represented as a uniform load instead of a point load. Eq. 77 and Eq. 78 apply to this situation using stringer spacing (in) instead of pile spacing and plank width in place of bent spacing.

$$W_{all} = \frac{b_{planking} h_{planking}^{\,2}}{0.6338 (stringer\,spacing)(b_{planking})} \sigma_{all,bend} \qquad \textbf{Eq. 83}$$

VI. Rapid Structural Assessment- Pier

Rapid Structural Assessment - Pier, or RSAP for short, is written in the C programming language (for code, see Appendix E). It is compiled on a PC to run in Windows 16-bit mode. The code is basic enough, however, that with a few minor changes, the program can run on a DOS platform. Additionally, portability to other platforms, such as UNIX, has been considered, and with a few changes, RSAP can be easily ported. However, the end users of this program, the UCTs, are using portable PCs in the field and thus, this program was customized to that end. The interface presents the user with simple query screens that prompt the user for data input. Once all of the inspection data has been entered, the user has the option to view the results on-screen, make changes, or pull up a previously entered file. When the user exits the program, the inspection data and the assessment results are printed to a text file that can be easily viewed with any word processor or text editor. Additionally, a script file is generated that can be run in MATLAB® which will provide graphical representation, such as that in Figure 19 of the assessment results. Included in the results section of the report are uniform loading and concentrated loading capacities for the inspected pier. The results will also identify various sized equipment such as forklifts, trucks, and containers that can be placed on the pier.

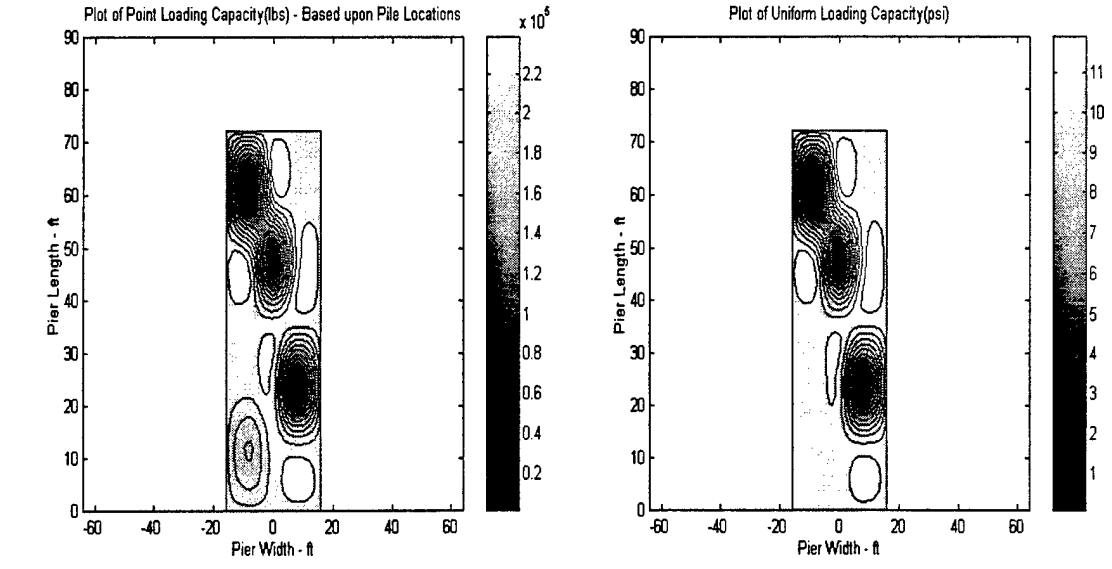


Figure 19. Sample MATLAB Output

VII. Conclusion

Now that we have calculated the allowable uniform capacity and concentrated load capacity for the main structural elements, we can establish the load capacity for the pier. In the case of the decking, the values will remain constant throughout. With the piles, however, the capacity will decrease as we move seaward on the pier due to the longer piles in deeper water. This may factor in to the ultimate pier capacity. If so, the results will reflect this because it is possible for a ship to off-load at various locations on a pier. If the lower value of capacity for the longer piles were established for the whole pier, it may unnecessarily restrict use of an otherwise capable pier. For the other three structural members, the lessor value will prevail as the load capacity for those members. This value will require adjustment for the dead load before we have our assessed capacity in the vertical direction.

We have also determined the allowable capacity in the lateral direction. After adjusting for environmental loads, we can determine the loading capacity per linear foot of pier that a ship may impose. With unit values for the various environmental loadings, an approximate ship size can be evaluated for safe berth at the pier.

The UCT's inspect piers that are being considered for use in tactical operations. They can usually perform an inspection on an average-sized pier in days. The results of their inspection, however, provide them no indication of the piers load carrying capability. This prevents the pier from being considered for any type of combat operation. RSAP enables the UCT inspection team obtain an estimate of the piers structural capacity.

Throughout the development of this assessment, all analytical calculations have been reduced to variables, or inputs, that are part of the inspection results already, or easily collectible on-site. Additionally, the assessment has been established using the guidelines for design of marine waterfront structures. When portions of the assessment were not readily quantifiable, assumptions were made that tended toward conservative, thus ensuring that the results would be within an acceptable range. This will enable a rapid, reliable, assessment of any timber pier being considered for tactical operations.

References

[1] Tsinker, G. P., 1997. *Handbook of Port and Harbor Engineering*. Chapman and Hall, New York.

[2] Underwater Construction Team 2 Web Page.

[3] Gaythwaite, J. W., 1990. *Design of Marine Facilities for the Berthing, Mooring, and Repair of Vessels*. Van Nostrand Reinhold, New York.

[4] NAVFAC (1980). *Piers and Wharves*. Piers and Wharves. Design Manual 25.1. Department of the Navy, Naval Facilities Engineering Command, NAVFAC DM-25.1, Alexandria, VA 22332-2300.

[5] NAVFAC(1994). Piers and Wharves. Piers and Wharves Military Handbook 1025/1. Department of the Navy, Naval Facilities Engineering Command, MIL-HDBK-1025/1, Change 3, Alexandria, VA 22332-2300.

[6] NAVFAC(1988). *General Criteria for Waterfront Construction*. Military Handbook 1025/6. Department of the Navy, Naval Facilities Engineering Command, MIL-HDBK-1025/6, Alexandria, VA 22332-2300.

[7] AASHTO(1989). *Standard Specifications for Highway Bridges*, 14th ed. Plus annual supplements to 1991. Washington, D.C.

[8] Conversation with Mr. Herb Herrmann, Deputy Director, Ocean Facilities Program, Naval Facilities Engineering Service Center, East Coast Detachment, Washington, D.C., 20374.

[9] NAVFAC (1997). *Conventional Underwater Construction and Repair Techniques*. Publication 990. Department of the Navy, Naval Facilities Engineering Command, NAVFAC P-990 for CD January 1997, Alexandria, VA 22332-2300.

[10] Coastal Engineering Research Center(1984). *Shore Protection Manual, Volume II*. Department of the Army, Waterways Experiment Station, Corps of Engineers, Vicksburg, MS, 39180.

[11] Beer, F. P. and E. R. Johnston, Jr. (1981). Mechanics of Materials, McGraw-Hill, New York

[12] Avallone, E. A. and T. Baumeister III, eds. (1986). *Marks' Standard Handbook for Mechanical Engineers*, 9th ed. McGraw-Hill, New York.

[13] Lindeburg, M. R. (1990). *Engineer-In-Training Reference Manual*, 7th ed. Professional Publications, Belmont, CA.

Appendix A - Levels of Inspection

3.3.1 Levels of Inspection

Three basic types or levels of inspection are used for inspecting marine facilities. They are distinguished by the resources and preparation needed to do the work and the type of damage/defect that is detectable, as:

• **Level I - General Visual Inspection.** The Level I effort can confirm as-built structural plans and detect obvious major damage or deterioration due to over-stress (collisions, ice), severe corrosion, or extensive biological growth and attack. This type of inspection does not involve cleaning of any structural elements and can therefore be conducted much more rapidly than the other types of inspections. The Level I effort is essentially a general inspection "swim-by" overview. It does not involve cleaning of structural elements, which allows the inspection to be conducted rapidly. The underwater inspector relies primarily on visual and/or tactile observations (depending on water clarity) to make condition assessments. These observations are made over the specified exterior surface area of the underwater structure, whether it is a quaywall, bulkhead, seawall, pile, or mooring. Although this is an overview, close attention should be given to confirming or providing information to update available facility drawings and condition evaluations.

• **Level II - Close-Up Visual Inspection.** Level II efforts are complete, detailed investigations of selected components or sub-components, or critical areas of the structure, directed toward detecting and describing damaged or deteriorated areas that may be hidden by surface bio-fouling. Limited deterioration measurements are obtained. These data are sufficient for gross estimates of facility load capability. This type of inspection will generally involve prior or concurrent cleaning of part of the structural elements. Since cleaning is time consuming, it is generally restricted to areas that are critical or that may be typical of the entire structure. Simple instruments such as calipers and measuring scales are commonly used to take physical measurements. Subjective judgments of structural integrity are occasionally made by probing wood with ice picks and by pounding concrete with hammers.

• **Level III - Highly Detailed Inspection.** This level of inspection is primarily designed to provide data that can be used to perform a structural assessment and will often require the use of Nondestructive Testing (NDT) techniques. The procedures are conducted to detect hidden or imminent damage. The training, cleaning, and testing requirements will vary depending on the type of damage/defect that is to be investigated and the type of inspection equipment to be used. In some cases, Level III inspections will require the use of partially destructive techniques such as sample coring in wood or concrete, material sampling, and in-situ surface hardness. The use of Level III inspection techniques is usually limited to key structural areas that may be suspect, or to structural areas that may be representative of the overall structure.

Appendix B - Pile Inspection Record

PILE INSPECTION RECORD

LOCATION | DATE | DIVERS

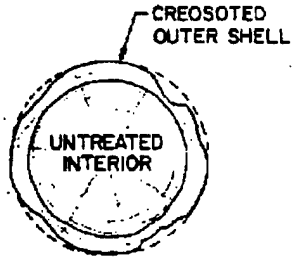
PIER NAME/NO. | PILE TYPE ☐ BEARING ☐ FENDER ☐ SHEET | PILE MATERIAL ☐ TIMBER ☐ STEEL ☐ REINFORCED CONCRETE

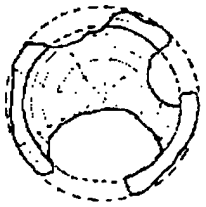
WATER DEPTH | TIME OF DAY | TIDE | DEPTH OF DAMAGE FROM DATUM = GUAGE DEPTH - TIDE

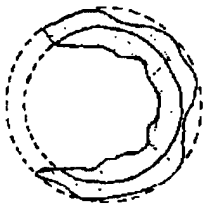
BENT NO.	PILE NO.	NI	PILE CONDITION					TYPE DAMAGE			GAUGE DEPTH DAMAGE	DIMENSIONS OF DAMAGE			COMMENTS
			ND	MN	MD	MJ	SV	MECH.	BIO	FUNC.		HGT	WIDTH	PENETR	

Figure 3-12. Standard pile inspection report form. (See Figures 3-13 and 3-14 for explanation of Ratings.)

Appendix C - Pile Condition Ratings for Timber Piles



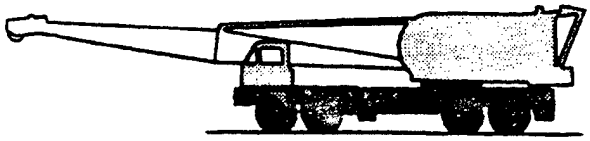




Timber Pile Condition Rating	Explanation
NI	Not inspected, inaccessible or passed by
ND	No defects: - less than 5% lost material - sound surface material - no evidence of borer damage
MN	Minor Defects: - 5 to 10% lost material - sound surface material - no evidence of borer damage - minor abrasion damage
MD	Moderate Defects: - 15 to 45% lost material - significant loss of outer shell material - evidence of borer damage - significant abrasion damage
MJ	Major Defects: - 45 to 75% lost material - significant loss of outer shell and interior material - evidence of severe borer damage - severe abrasion damage
SV	Severe Defects - more than 75% lost material - no remaining structural strength - severe borer damage

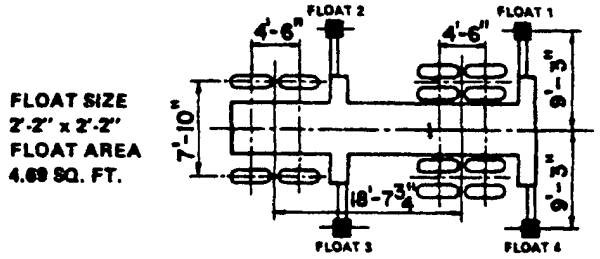
Figure 3-14.
Explanation of pile condition ratings for timber piles.

Appendix D - Crane Loading Data Charts



50-TON TRUCK CRANE

CRANE TRAVELING – BOOM OVER FRONT



FLOAT SIZE
2'-2" x 2'-2"
FLOAT AREA
4.69 SQ. FT.

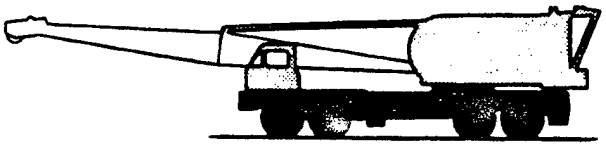
1. DATA SHOWN FOR OUTRIGGER FLOAT LOADS ARE FOR OVER SIDE AND OVER REAR LIFTS. FOR OVER FRONT LIFTS, A FRONT BUMPER FLOAT IS REQUIRED.
2. BOOM IS OVER THE CORNER FOR WHICH FLOAT LOAD IS GIVEN.
3. FOR EQUAL RADII, RATED LOADS VARY ACCORDING TO BOOM LENGTH.

PLAN – OUTRIGGERS EXTENDED

CRANE TRAVELING – RUBBER TIRE WHEEL LOADS (LBS)						
TIRES	BOOM LENGTH	TOTAL WEIGHT	BOOM OVER FRONT		BOOM OVER REAR	
NO. SIZE	(FT.)	(LBS.)	EACH FRONT SINGLE TIRE	EACH REAR DUAL TIRE	EACH FRONT SINGLE TIRE	EACH REAR DUAL TIRE
12 14.00-20	40	107,566	5,024	21,868	10,164	16,727

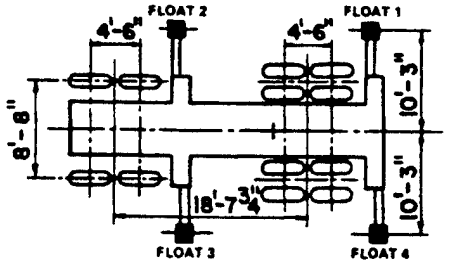
OUTRIGGER FLOAT LOADS (LBS)						
RATED LOAD (LBS.)	RADIUS (FT.)	BOOM LENGTH (FT.)	FLOAT NUMBER (1)	(2)	(3)	(4)
100,000	12	40	89,500	107,000	112,000	84,500
59,800	25	40	89,500	107,000	112,000	84,500
46,700	30	40	85,000	101,700	106,400	80,300
30,700	40	40	75,200	89,100	94,100	71,000
22,900	50	50	72,500	86,700	90,700	68,500
17,700	60	60	69,800	83,500	87,400	65,900
14,500	70	70	↓	↓	↓	↓
11,800	80	80				
9,700	90	90				
8,050	100	100				
6,750	110	110				
5,550	120	120				
4,550	130	130				
4,150	130	140				
3,900	130	150	69,800	83,500	87,400	65,900

FIGURE 9
50-Ton Truck Crane Loadings
25.1-27



70-TON TRUCK CRANE

CRANE TRAVELING – BOOM OVER FRONT



FLOAT SIZE
2'-2" x 2'-2"
FLOAT AREA
4.69 SQ. FT.

PLAN – OUTRIGGERS EXTENDED

1. DATA SHOWN FOR OUTRIGGER FLOAT LOADS ARE FOR OVER SIDE AND OVER REAR LIFTS. FOR OVER FRONT LIFTS, A FRONT BUMPER FLOAT IS REQUIRED.
2. BOOM IS OVER THE CORNER FOR WHICH FLOAT LOAD IS GIVEN.
3. FOR EQUAL RADII, RATED LOADS VARY ACCORDING TO BOOM LENGTH.

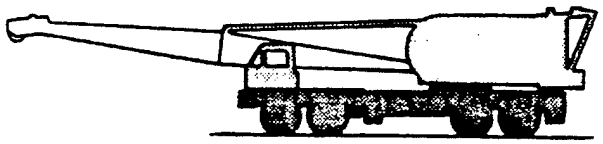
CRANE TRAVELING – RUBBER TIRE WHEEL LOADS (LBS)

TIRES NO.	TIRES SIZE	BOOM LENGTH (FT.)	TOTAL WEIGHT (LBS.)	BOOM OVER FRONT EACH FRONT SINGLE TIRE	BOOM OVER FRONT EACH REAR DUAL TIRE	BOOM OVER REAR EACH FRONT SINGLE TIRE	BOOM OVER REAR EACH REAR DUAL TIRE
12	14.00-20	40	114,580	4,683	23,963	12,780	15,865

OUTRIGGER FLOAT LOADS (LBS)

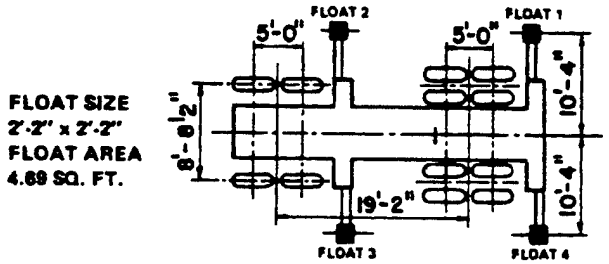
RATED LOAD (LBS.)	RADIUS (FT.)	BOOM LENGTH (FT.)	FLOAT NUMBER (1)	(2)	(3)	(4)
140,000	12	40	121,000	145,500	151,000	115,000
107,000	20	40	121,000	145,500	151,000	115,000
59,100	30	40	100,400	120,800	125,300	95,500
37,300	40	40	87,100	104,800	108,700	82,800
27,800	50	50	81,700	98,200	102,000	77,600
21,500	60	60	↕	↕	↕	↕
17,100	70	70				
14,000	80	80				
11,500	90	90				
9,550	100	100				
8,150	110	110				
6,800	120	120				
5,600	130	130				
4,600	140	140				
3,650	150	150				
3,400	150	160				
3,150	150	170				
2,900	150	180	81,700	98,200	102,000	77,000

FIGURE 10
70-Ton Truck Crane Loadings
25.1-28



CRANE TRAVELING – BOOM OVER FRONT

90-TON TRUCK CRANE



FLOAT SIZE
2'-2" x 2'-2"
FLOAT AREA
4.69 SQ. FT.

PLAN – OUTRIGGERS EXTENDED

1. DATA SHOWN FOR OUTRIGGER FLOAT LOADS ARE FOR OVER SIDE AND OVER REAR LIFTS. FOR OVER FRONT LIFTS, A FRONT BUMPER FLOAT IS REQUIRED.
2. BOOM IS OVER THE CORNER FOR WHICH FLOAT LOAD IS GIVEN.
3. FOR EQUAL RADII, RATED LOADS VARY ACCORDING TO BOOM LENGTH.

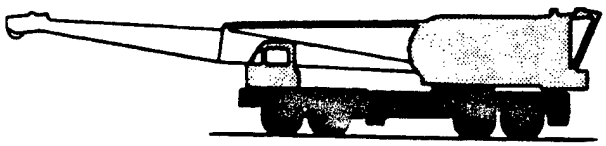
CRANE TRAVELING – RUBBER TIRE WHEEL LOADS (LBS)

TIRES NO.	TIRES SIZE	BOOM LENGTH (FT.)	TOTAL WEIGHT (LBS.)	BOOM OVER FRONT EACH FRONT SINGLE TIRE	BOOM OVER FRONT EACH REAR DUAL TIRE	BOOM OVER REAR EACH FRONT SINGLE TIRE	BOOM OVER REAR EACH REAR DUAL TIRE
12	14.00-24	50	136,789	6,107	28,091	12,115	22,083

OUTRIGGER FLOAT LOADS (LBS)

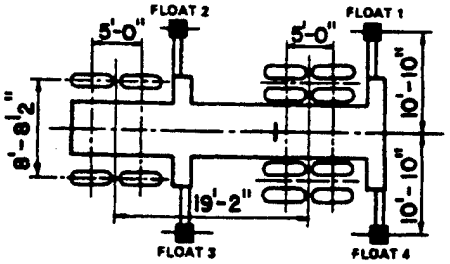
RATED LOAD (LBS.)	RADIUS (FT.)	BOOM LENGTH (FT.)	FLOAT NUMBER (1)	(2)	(3)	(4)
180,000	12	50	133,500	180,500	187,000	126,500
125,000	20	↕	133,500	180,500	187,000	126,500
71,800	30	↕	114,800	155,200	160,800	108,800
46,200	40	↕	100,100	135,400	140,300	94,900
33,500	50	50	93,450	126,350	130,900	88,550
25,900	60	60	↕	↕	↕	↕
20,700	70	70	↕	↕	↕	↕
16,900	80	80	↕	↕	↕	↕
14,000	90	90	↕	↕	↕	↕
11,600	100	100	↕	↕	↕	↕
9,700	110	110	↕	↕	↕	↕
8,350	120	120	↕	↕	↕	↕
7,000	130	130	↕	↕	↕	↕
5,750	140	140	↕	↕	↕	↕
4,700	150	150	↕	↕	↕	↕
4,400	↕	160	↕	↕	↕	↕
4,200	↕	170	↕	↕	↕	↕
3,950	↕	180	↕	↕	↕	↕
3,600	↕	190	↕	↕	↕	↕
3,300	150	200	93,450	126,350	130,900	88,550

FIGURE 11
90-Ton Truck Crane Loadings
25.1-29



115-TON TRUCK CRANE

CRANE TRAVELING – BOOM OVER FRONT



FLOAT SIZE
2'-2" x 2'-2"
FLOAT AREA
4.69 SQ. FT.

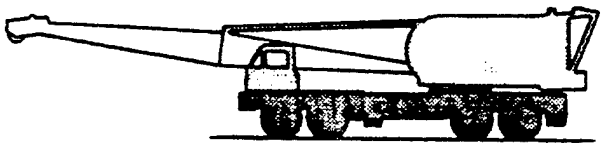
1. DATA SHOWN FOR OUTRIGGER FLOAT LOADS ARE FOR OVER SIDE AND OVER REAR LIFTS. FOR OVER FRONT LIFTS, A FRONT BUMPER FLOAT IS REQUIRED.
2. BOOM IS OVER THE CORNER FOR WHICH FLOAT LOAD IS GIVEN.
3. FOR EQUAL RADII, RATED LOADS VARY ACCORDING TO BOOM LENGTH.

PLAN – OUTRIGGERS EXTENDED

CRANE TRAVELING – RUBBER TIRE WHEEL LOADS (LBS)						
			BOOM OVER FRONT		BOOM OVER REAR	
TIRES NO. SIZE	BOOM LENGTH (FT.)	TOTAL WEIGHT (LBS.)	EACH FRONT SINGLE TIRE	EACH REAR DUAL TIRE	EACH FRONT SINGLE TIRE	EACH REAR DUAL TIRE
12 14.00-24	50	163,190	4,521	36,277	15,518	25,280

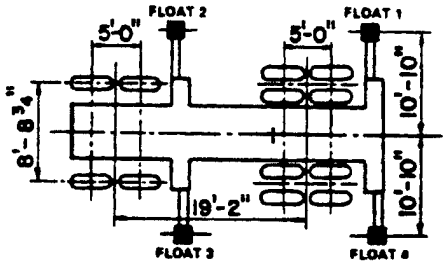
OUTRIGGER FLOAT LOADS (LBS)						
RATED LOAD (LBS.)	RADIUS (FT.)	BOOM LENGTH (FT.)	FLOAT NUMBER (1)	(2)	(3)	(4)
230,000	12	50	185,000	233,000	241,500	176,500
161,100	20	↑	185,000	233,000	241,500	176,500
78,900	30	↕	138,800	174,800	181,100	132,400
50,900	40	↓	118,400	149,100	154,600	113,000
37,000	50	50	111,000	139,800	144,900	105,900
28,500	60	60	↑	↑	↑	↑
22,900	70	70	↕	↕	↕	↕
18,900	80	80	↕	↕	↕	↕
15,600	90	90	↕	↕	↕	↕
13,000	100	100	↕	↕	↕	↕
10,800	110	110	↕	↕	↕	↕
9,200	120	120	↕	↕	↕	↕
7,700	130	130	↕	↕	↕	↕
6,400	140	140	↕	↕	↕	↕
5,200	150	150	↕	↕	↕	↕
4,200	160	160	↕	↕	↕	↕
3,250	170	170	↕	↕	↕	↕
2,950	170	180	↕	↕	↕	↕
2,500	170	190	↕	↕	↕	↕
2,250	170	200	111,000	139,800	144,900	105,900

FIGURE 12
115-Ton Truck Crane Loadings
25.1-30



140-TON TRUCK CRANE

CRANE TRAVELING – BOOM OVER FRONT



FLOAT SIZE
2'-2" x 2'-2"
FLOAT AREA
4.69 SQ. FT.

1. DATA SHOWN FOR OUTRIGGER FLOAT LOADS ARE FOR OVER SIDE AND OVER REAR LIFTS. FOR OVER FRONT LIFTS, A FRONT BUMPER FLOAT IS REQUIRED.
2. BOOM IS OVER THE CORNER FOR WHICH FLOAT LOAD IS GIVEN.
3. FOR EQUAL RADII, RATED LOADS VARY ACCORDING TO BOOM LENGTH.

PLAN – OUTRIGGERS EXTENDED

CRANE TRAVELING – RUBBER TIRE WHEEL LOADS (LBS)

TIRES NO.	TIRES SIZE	BOOM LENGTH (FT.)	TOTAL WEIGHT (LBS.)	BOOM OVER FRONT EACH FRONT SINGLE TIRE	BOOM OVER FRONT EACH REAR DUAL TIRE	BOOM OVER REAR EACH FRONT SINGLE TIRE	BOOM OVER REAR EACH REAR DUAL TIRE
12	14.00-24	50	168,821	6,711	35,494	13,759	28,447

OUTRIGGER FLOAT LOADS (LBS)

RATED LOAD (LBS.)	RADIUS (FT.)	BOOM LENGTH (FT.)	FLOAT NUMBER (1)	(2)	(3)	(4)
280,000	12	50	172,500	225,500	233,500	164,000
138,360	25	↑	172,500	225,500	233,500	164,000
114,400	30	↓	163,900	214,200	221,800	155,800
74,900	40	↓	146,600	191,700	198,500	139,400
55,100	50	50	133,700	174,800	181,000	127,100
43,400	60	60	131,100	171,400	177,500	124,600
35,100	70	70	↑	↑	↑	↑
29,200	80	80	↓	↓	↓	↓
25,200	90	90	↓	↓	↓	↓
21,400	100	100	↓	↓	↓	↓
18,500	110	110	↓	↓	↓	↓
15,900	120	120	↓	↓	↓	↓
14,000	130	130	↓	↓	↓	↓
12,200	140	140	↓	↓	↓	↓
10,500	150	150	↓	↓	↓	↓
9,050	160	160	↓	↓	↓	↓
7,700	170	170	↓	↓	↓	↓
6,650	180	180	↓	↓	↓	↓
5,550	190	190	↓	↓	↓	↓
4,500	200	200	↓	↓	↓	↓
4,300	200	210	↓	↓	↓	↓
3,500	200	240	↓	↓	↓	↓
2,650	200	270	131,100	171,500	177,500	124,600

FIGURE 13
140-Ton Truck Crane Loadings
25.1-31

Appendix E - Program Listing

```
/* (C)1999 All rights reserved - R. J. Keiter */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <ctype.h>
#include <dos.h>
#include <conio.h>
#include <time.h>
#include <io.h>
#include <fcntl.h>
#include <process.h>
#include <sys\stat.h>

#define PI 3.141592654
#define GV 32.174
#define MIN_FOS 1.5
#define MAX_FOS 5.0
#define H2O_RG 64.0

/*********************************************************/

typedef struct _pile Pile;
struct _pile
{
    char id_let[3];
    double diameter;
    double length_mud_to_cap;
    double unsup_length;
    double eff_length;
    char condition[3];
    int flag;
    double vert_load;
    double uni_load_psi;
    double lat_load;
    double dead_load;
    double spring_cap;
    double load_area;
    double pile_weight;
    Pile *next;
    Pile *prev;
};

typedef struct _bent Bent;
struct _bent
{
    int id_num;
    double depth;
    double freeboard;
    double pile_length;
```

```
    double brace_2_cap;
    double pile_length_tot;
    double spring_tot;
    double weight_tot;
    double dead_wt_per_pile;
    double dead_wt;
    double dead_temp;
    double load_area;
    double lat_cap;
    char batters;
    Pile *pile;
    Bent *next;
};

typedef struct _props Props;
struct _props
{
    double bending_mod;
    double sig_pub_comp;
    double sig_pub_bend;
    double unit_wt;
    char tim_type[25];
};

typedef struct _loads Load;
struct _loads
{
    char fork_lim_5;
    char fork_lim_6;
    char fork_lim_75;
    char fork_lim_8;
    char fork_lim_10;
    char fork_lim_12;
    char fork_lim_15;
    char fork_lim_20;
    double hs_limit_wheel_str;
    double h_limit_wheel_str;
    double string_pt_lim;
    double string_uni_lim_psi;
    double axle_lim_cap;
    double cap_pt_lim;
    double cap_uni_lim_psi;
    double plank_uni_lim_psi;
    double pile_pt_miss;
    double pile_uni_miss;
    double pile_uni_min;
    double pile_pt_miss_end;
    double pile_uni_miss_end;
};

typedef struct _env_data Env;
struct _env_data
{
```

```
    double wind_spd;
    double wind_spd_ht;
    double wind_10;
    double wind_angle;
    double wind_angle_deg;
    double current_spd;
    double current_angle;
    double current_angle_deg;
    double current_pile_CD;
    double current_Re;
    double fouling_factor;
    char soil_type[25];
    int soil_flag;
};

typedef struct _wave_data Wave;
struct _wave_data
{
    double wave_H;
    double wave_T;
    double wave_d;
    double wave_angle;
    double wave_angle_deg;
    double wave_Re;
    double wave_k;
    double wave_c;
    double wave_Ki_max;
    double wave_KD_max;
    double wave_Si;
    double wave_SD;
    double lambda_A;
    double lambda_0;
    double wave_umax;
    double wave_CD;
    double wave_CM;
    double wave_M_max_d;
    char wave_Airy;
};

typedef struct _batters Batters;
struct _batters
{
    char batters;
    double batter_size;
    double batter_rise;
    double batter_run;
    double batter_angle;
    double batter_angle_deg;
};

typedef struct _sup_struct Sup_struct;
struct _sup_struct
{
```

```
    Props *sup_struct_prop;
    char cap_type;
    double cap_ht;
    double cap_w;
    double cap_wt;
    double string_spread;
    double string_ht;
    double string_w;
    double string_wt;
    double string_wt_bent;
    double plank_ht;
    double plank_w;
    double plank_wt;
    char bracing;
};

typedef struct _flagz Flagz;
struct _flagz
{
    char flag;
    int s_flag;
    int t_flag;
    int flag1;
    int flag2;
    int flag3;
    int flag4;
    int flag5;
};

typedef struct _pier_forces Force;
struct _pier_forces
{
    double wind;
    double wind_x;
    double wind_70;
    double current_ship;
    double current_ship_x;
    double current_pier_M;
    double current_pier_M_x;
    double wave_M_max_d;
    double wave_M_x_d;
};

typedef struct _ship_data Ship;
struct _ship_data
{
    double length;
    double C1;
    double C2;
    double draft;
    double freeboard;
    double force_tot;
};
```

```
typedef struct _pier_data PierData;
struct _pier_data
{
    int no_of_bents;
    int piles_per_bent;
    int ship;
    double factor_of_safety;
    double pile_size;
    double fixity;
    double bent_spread;
    double pile_spread;
    double pier_length;
    double pier_width;
    double spring_tot;
    double weight_tot;
    double pier_period;
    double lat_cap;
    double lat_cap_per_ft;
    double est_ton_ship;
    double pile_uni_min;
    Props *pile_prop;
    Force *forces;
    Load *loads;
    Env *env_data;
    Bent *bents;
  Wave *wave_data;
    Sup_struct *sup_struct;
    Batters *batters;
    Ship *ship_data;
};

/*********************************************************/

void clean(void);
void open_msg(void);
void exit_msg(Flagz *);
void file_msg(Flagz *);
void t_check(PierData *, Flagz *);
void date_plot(FILE *);
void pause(void);
void put_date(void);
void menu (void);
void get_pier_data(PierData *);
void get_ship_data(PierData *);
void get_prop_info(Props *, int);
void get_bent_info(PierData *);
void get_pile_info(Bent *, PierData *);
void get_cond(Bent *, Pile *, PierData *, int);
void get_env_info(PierData *);
void get_soil_info(Env *);
void get_load_info(PierData *);
void prn_to_screen(void);
```

```
void pile_plot(PierData *, char);
void prt_err(char[35]);
void prt_inv(void);
char prn_to_file(PierData *, Flagz *);
int quit(char);
int test_input_int(int);
int check_num(void);
double test_input_double(double);
double is_valid(double, double);
double test_input(int, double);
char check_fork(double, double, PierData *);

/*********************************************************/

int main(void)
{
    menu();
    return(0);
}

/*********************************************************/

void menu (void)
{
    char choice = '0';
    PierData *input;
    Flagz *flagz;

    put_date();
    open_msg();
    clean();
    input = calloc(1,sizeof(PierData));

    if(input == NULL)
        {
            prt_err("PierData");
        }

    flagz = calloc(1,sizeof(Flagz));

    if(flagz == NULL)
        {
            prt_err("Flags");
        }

    while(choice != '3')
        {
            clean();
            flagz->flag = 'Q';
            printf("\n\n\t\tRSAP - Rapid Structural Assessment, Pier v1.0");
            printf("\n\n\t\t\tDisplay File Menu");
            printf("\n\n\t\t\t1) Input Inspection Data");
            printf("\n\t\t\t2) Display File");
```

```
            printf("\n\t\t\t3) Exit");
            printf("\n\n\t\t\tEnter selection: ");
            choice = getche();

            switch(choice)
                {
                    case '1':
                        fflush(stdin);
                        clean();
                        t_check(input, flagz);
                        get_pier_data(input);
                        flagz->t_flag = 1;
                        flagz->flag = prn_to_file(input, flagz);
                        flagz->flag1 = 1;
                        break;

                    case '2':
                        fflush(stdin);
                        clean();
                        prn_to_screen();
                        break;

                    case '3':
                        if(flagz->flag1 == 1||flagz->flag4 == 1)
                            {
                                fflush(stdin);
                                flagz->t_flag = 0;
                                flagz->flag = prn_to_file(input, flagz);
                                exit_msg(flagz);
                                choice = '3';
                            }

                        else
                            {
                                fflush(stdin);
                                exit_msg(flagz);
                                choice = '3';
                            }
                        break;

                    default:
                        printf("\n\n\tInput should be 1,2,3,...");
                        prt_inv();
                        break;
                }
        }

    free(input);
}

/*********************************************************/

void open_msg(void)
```

```
{
    printf("\n\n\n\tWelcome to RSAP - Rapid Structural Assessment, Pier. This");
    printf("\n\tprogram provides a means for rapidly estimating the capacity");
    printf("\n\tof a timber, open, pile supported pier. It requires input that");
    printf("\n\tis gathered by the US Navy Underwater Construction Teams during");
    printf("\n\ta waterfront inspection. When prompted for input regarding");
    printf("\n\tmeasurements, pay close attention to what is being asked for.");
    printf("\n\tWhen entering dimension information, the height and width are");
    printf("\n\tthe vertical and horizontal dimensions of the member as it is");
    printf("\n\tplaced. For example, planking is layed on its wide side. Thus");
    printf("\n\tthe \"height\" is the narrow dimension. When prompted to select");
    printf("\n\tfrom a list of choices, you need only enter the choice. You do");
    printf("\n\tNOT need to press \"enter\" after selecting your choice. ");
    printf("\n");
    printf("\n\t\(C)1999 - All rights reserved. R. J. Keiter");
    printf("\n\n\t\Questions? Problems? Contact me at: rjkeiter@iname.com");
    printf("\n");
    printf("\n");
    pause();
}

/*********************************************************/

void exit_msg(Flagz *flagz)
{
    clean();
    printf("\n");
    printf("\n\tThank you for using RSAP - Rapid Structural Assessment, Pier.");
    printf("\n\n");
    file_msg(flagz);
}

/*********************************************************/

void file_msg(Flagz *flagz)
{
    char filez[3][13];
    int i, j;

    fflush(stdin);
    printf("\n");

    if(flagz->flag != 'Q')
        {
            strcpy(filez[0],"pile_in0.dat");
            j = 1;

            if(flagz->s_flag == 1)
                {
                    strcpy(filez[1],"pil_plt0.m");
                    strcpy(filez[2],"uni_plt0.m");
                    j = 3;
                }
```

```
            printf("\n\tThe following file(s) have been saved to your ");
            printf("working directory:");
            printf("\n");

            for(i = 0; i < j; i++)
                {
                    filez[i][7] = flagz->flag;
                    printf("\n\t\t\t%s", filez[i]);
                }
        }
    printf("\n\n");
    pause();
}

/*********************************************************/

void t_check(PierData *input, Flagz *flagz)
{
    char y_n;

    if(flagz->flag1 == 1)
        {
            printf("\n\n\tDo you wish to save the data you have");
            printf(" already input(y/n)? ");
            y_n = getche();

            while(y_n != 'y'&&y_n != 'Y'&&y_n != 'n'&&y_n != 'N')
                {
                    prt_inv();
                    printf("\n\n\tDo you wish to save your data (y/n)?: ");
                    y_n = getche();
                }

            if(y_n == 'y'||y_n == 'Y')
                {
                    flagz->t_flag = 0;
                    flagz->flag = prn_to_file(input, flagz);
                    file_msg(flagz);
                }
        }
}

/*********************************************************/

void get_pier_data(PierData *Input)
{
    Wave *wtemp;
    Force *ftemp;
    char batters, cap_type, bracing, choice, choice1, f_o_str[20];
    double batter_rise_ft, batter_run_ft, bent_spread_ft, pile_spread_ft;
    double batter_run_in, bent_spread_in, batter_rise_in, pile_spread_in;
    double str_spr_ft, str_spr_in, CD_ship = 1.0, bio_Re, c_speed;
```

```
double f_o_s = 0, j = 0, I_pile, EI_pile, t, theta, gamma, wave_Ki, wave_KD;
double cwtemp, wave_Mi, wave_Mia, wave_Mib, wave_Mic;
double wave_MD, wave_MDa, wave_MDb, wave_MDc;
int i;

clean();
get_env_info(Input);
clean();
wtemp = Input->wave_data;
ftemp = Input->forces;
printf("\n\tEnter the factor of safety you");
printf("\n\twish to use (recommended: 3, min");
printf(" = %.2lf): ", MIN_FOS);
scanf(" %s", &f_o_str);

while(j < MIN_FOS||j > MAX_FOS)
   {
      f_o_s = j = atof(f_o_str);

      if(f_o_s < MIN_FOS)
         {
            printf("\n\tInvalid input! The minimum factor of safety");
            printf(" is %.2lf!", MIN_FOS);
            printf("\n\tRe-enter value: ");
            scanf(" %s", &f_o_str);
         }

      if(f_o_s > MAX_FOS)
         {
            printf("\n\tAre sure you want this high of a");
            printf("\n\tfactor of safety (y/n)? ");
            choice = getche();

            while(choice != 'y'&&choice != 'Y'&&choice != 'n'&&choice != 'N')
               {
                  prt_inv();
                  printf("\n\n\tAre sure you want this high of a");
                  printf("\n\tfactor of safety (y/n)? ");
                  choice = getche();
               }

            if(choice == 'n'||choice == 'N')
               {
                  printf("\n\tRe-enter value: ");
                  scanf(" %s", &f_o_str);
               }
            else
               j = 3;
         }
   }

Input->factor_of_safety = f_o_s;
clean();
```

```
printf("\n\tEnter number of bents: ");
Input->no_of_bents = test_input_int(1);
printf("\n\tEnter distance between bents,");
printf("\n\t\tfeet: ");
bent_spread_ft = test_input_double(0.0);
printf("\t\tinches: ");
bent_spread_in = test_input_double(0.0);
Input->bent_spread = bent_spread_ft + bent_spread_in/12;
Input->pier_length = Input->no_of_bents * Input->bent_spread;
clean();
printf("\n\tEnter piling diameter size(inches): ");
Input->pile_size = test_input_double(1.0);
clean();

if(wtemp->wave_H > 0.0)
   {
      wtemp->wave_Re = wtemp->wave_umax*100000*Input->pile_size/12;

      if(wtemp->wave_Re < 250000)
         wtemp->wave_CM = 2.0;
      else
         if(wtemp->wave_Re < 500000)
            wtemp->wave_CM = (2.5 - wtemp->wave_Re/500000);
         else
            wtemp->wave_CM = 1.5;

      if(wtemp->wave_Re < 300000)
         wtemp->wave_CD = 1.2;
      else
         wtemp->wave_CD = 0.6;

      for(i = 0; i < 361; i++)
         {
            t = wtemp->wave_T*i/360;
            theta = 2*PI*wtemp->wave_d/wtemp->lambda_A;
            gamma = 2*PI*t/wtemp->wave_T;
            wave_Ki = 0.5*tanh(theta)*sin(-gamma);

            if(wave_Ki > wtemp->wave_Ki_max)
               wtemp->wave_Ki_max = wave_Ki;

            wave_KD = ((1+(2*theta)/(sinh(2*theta)))*fabs(cos(gamma))*cos(gamma))/8;

            if(wave_KD > wtemp->wave_KD_max)
               wtemp->wave_KD_max = wave_KD;

            wave_Mia = wtemp->wave_CM*PI*pow((Input->pile_size/12),2);
            wave_Mib = wtemp->wave_H*H2O_RG*wave_Ki/4;
            wave_Mic = wtemp->wave_Si*wtemp->wave_d;
            wave_Mi = wave_Mia*wave_Mib*wave_Mic;
            wave_MDa = wtemp->wave_CD*0.5*(Input->pile_size/12);
            wave_MDb = pow(wtemp->wave_H,2)*H2O_RG*wave_KD;
            wave_MDc = wtemp->wave_SD*wtemp->wave_d;
```

```
                wave_MD = wave_MDa*wave_MDb*wave_MDc;

                if(((wave_Mi + wave_MD)/wtemp->wave_d) > wtemp->wave_M_max_d)
                    ftemp->wave_M_max_d = (wave_Mi + wave_MD)/wtemp->wave_d;
            }
        }

    wtemp->wave_M_max_d = ftemp->wave_M_max_d;
    ftemp->wave_M_x_d = (ftemp->wave_M_max_d*sin(wtemp->wave_angle));

    if(Input->env_data->current_spd > 0.0)
        {
            c_speed = Input->env_data->current_spd*5280/3600;
            Input->env_data->current_Re = c_speed*(Input->pile_size/12)*100000;

            if(Input->env_data->current_Re < 300000)
                Input->env_data->current_pile_CD = 1.2;
            else
                Input->env_data->current_pile_CD = 0.6;

            ftemp->current_ship = ((H2O_RG/GV)*pow(c_speed,2)*CD_ship)/2;
            ftemp->current_ship_x = (ftemp->current_ship*sin(Input->env_data->current_angle));

            if(Input->env_data->fouling_factor > 0.0)
                {
                    bio_Re = (Input->pile_size+Input->env_data->fouling_factor/2)/12*100000;
                    Input->env_data->current_Re = c_speed * bio_Re;

                    if(Input->env_data->current_Re < 300000)
                        Input->env_data->current_pile_CD = 1.2;
                    else
                        Input->env_data->current_pile_CD = 0.6;
                }

            ftemp->current_pier_M = H2O_RG*Input->pile_size/12*pow(c_speed,2)*Input->env_data-
>current_pile_CD/4/GV;
            ftemp->current_pier_M_x = fabs(ftemp->current_pier_M*sin(Input->env_data->current_angle));
        }

    printf("\n\tWhat type of timber are the pilings?");
    Input->pile_prop = calloc(1, sizeof(Props));

    if(Input->pile_prop == NULL)
        {
            prt_err("Material Properties");
        }

    get_prop_info(Input->pile_prop, 1);
    I_pile = (PI*pow(Input->pile_size/2,4)/4);
    EI_pile = (I_pile*Input->pile_prop->bending_mod);

    if(Input->env_data->soil_flag == 0)
        Input->fixity = (8.5*Input->pile_size)/12;
```

```
else
    if(Input->env_data->soil_flag == 1)
        {
            if(EI_pile <= 10000000000)
                Input->fixity = 10;
            else
                Input->fixity = 12;
        }
    else
        if(Input->env_data->soil_flag == 2)
            {
                if(EI_pile <= (10000000000))
                    Input->fixity = 8;
                else
                    Input->fixity = 10;
            }
        else
            if(Input->env_data->soil_flag == 3)
                Input->fixity = 5;

clean();
printf("\n\n\tEnter number of pilings per bent: ");
Input->piles_per_bent = test_input_int(1);
fflush(stdin);

if(Input->piles_per_bent < 4)
    {
        clean();
        printf("\n\t*************************WARNING*************************");
        printf("\n\n\t  The accuracy of this assessment program is better for");
        printf("\n\t       a pier with more than %d piles per bent!!!",Input->piles_per_bent);
        printf("\n\n\t*************************WARNING*************************");
        printf("\n");
        pause();
clean();
    }

if(Input->piles_per_bent > 1)
    {
        printf("\n\tEnter distance between pilings in bent,");
        printf("\n\t\tfeet: ");
        pile_spread_ft = test_input_double(0.0);
        printf("\t\tinches: ");
        pile_spread_in = test_input_double(0.0);
        Input->pile_spread = pile_spread_ft+pile_spread_in/12;
    }

Input->pier_width = (Input->piles_per_bent-1) * Input->pile_spread;
clean();
Input->batters = calloc(1, sizeof(Batters));

if(Input->batters == NULL)
    {
```

```
        prt_err("Batters");
    }

printf("\n\tAre there batter pilings (y/n)?: ");
batters = getche();
printf("\n");

while(batters != 'y'&&batters != 'Y'&&batters != 'n'&&batters != 'N')
    {
        prt_inv();
        printf("\n\n\tAre there batter pilings (y/n)?: ");
        batters = getche();
        printf("\n");
    }

Input->batters->batters = batters;

if(batters == 'y'||batters == 'Y')
    {
        printf("\n\tEnter batter piling diameter size(inches): ");
        Input->batters->batter_size = test_input_double(1.0);
        printf("\n\tTo find the slope of the batter pile,");
        printf("\n\tplease enter the following information:");
        printf("\n\n\t\tVertical rise - feet: ");
        batter_rise_ft = test_input_double(0.0);
        printf("\t\tVertical rise - inches: ");
        batter_rise_in = test_input_double(0.0);
        Input->batters->batter_rise = batter_rise_ft + batter_rise_in/12;
        printf("\n\t\tHorizontal run - feet: ");
        batter_run_ft = test_input_double(0.0);
        printf("\t\tHorizontal run - inches: ");
        batter_run_in = test_input_double(0.0);
        Input->batters->batter_run = batter_run_ft + batter_run_in/12;

        if(Input->batters->batter_run == 0.0)
            Input->batters->batter_angle = PI/2;
        else
            Input->batters->batter_angle = atan(Input->batters->batter_rise/Input->batters->batter_run);

        Input->batters->batter_angle_deg = Input->batters->batter_angle*180/PI;
    }

clean();
printf("\n\tWhat type of timber is the superstructure?");
Input->sup_struct = calloc(1, sizeof(Sup_struct));

if(Input->sup_struct == NULL)
    {
        prt_err("Superstucture");
    }

Input->sup_struct->sup_struct_prop = calloc(1, sizeof(Props));
```

```
    if(Input->sup_struct->sup_struct_prop == NULL)
        {
            prt_err("Superstucture Properties");
        }

    get_prop_info(Input->sup_struct->sup_struct_prop, 0);
    clean();
    printf("\n\tIs there cross bracing on the pilings (y/n)?: ");
    bracing = getche();
    printf("\n");

    while(bracing != 'y'&&bracing != 'Y'&&bracing != 'n'&&bracing != 'N')
        {
            prt_inv();
            printf("\n\n\tIs there cross bracing (y/n)?: ");
            bracing = getche();
            printf("\n");
        }

    Input->sup_struct->bracing = bracing;
    clean();
    printf("\n\tIs the pile cap (S)olid or S(p)lit (\"s\" or \"p\")? ");
    cap_type = getche();
    printf("\n");

    while(cap_type != 's'&&cap_type != 'S'&&cap_type != 'p'&&cap_type != 'P')
        {
            prt_inv();
            printf("\n\n\tIs the pile cap (S)olid or S(p)lit (\"s\" or \"p\")? ");
            cap_type = getche();
            printf("\n");
        }

    Input->sup_struct->cap_type = cap_type;

    if(Input->sup_struct->cap_type == 's'||Input->sup_struct->cap_type == 'S')
        printf("\n\tEnter pile cap dimensions");
    else
        printf("\n\tEnter pile cap member dimensions");

    printf("\n\t\theight(inches): ");
    Input->sup_struct->cap_ht = test_input_double(1.0);
    printf("\t\twidth(inches): ");
    Input->sup_struct->cap_w = test_input_double(1.0);
    cwtemp = Input->sup_struct->cap_w/12;
    Input->sup_struct->cap_wt = Input->sup_struct->cap_ht/12*cwtemp*Input->pier_width*Input->sup_struct-
>sup_struct_prop->unit_wt;

    if(Input->sup_struct->cap_type == 'p'||Input->sup_struct->cap_type == 'P')
        {
            Input->sup_struct->cap_wt *= 2;
            cwtemp = Input->pile_size/12;
        }
```

```
    printf("\n\tEnter stringer dimensions");
    printf("\n\t\theight(inches): ");
    Input->sup_struct->string_ht = test_input_double(1.0);
    printf("\t\twidth(inches): ");
    Input->sup_struct->string_w = test_input_double(1.0);
    Input->sup_struct->string_wt = Input->sup_struct->string_ht/12*Input->sup_struct->string_w/12*(Input-
>bent_spread+cwtemp)*Input->sup_struct->sup_struct_prop->unit_wt;
    printf("\n\tEnter stringer spread - center to center");
    printf("\n\t\tfeet: ");
    str_spr_ft = test_input_double(0.0);
    printf("\t\tinches: ");
    str_spr_in = test_input_double(0.0);
    Input->sup_struct->string_spread = str_spr_ft + str_spr_in/12;
    Input->sup_struct->string_wt_bent = (Input->pier_width/Input->sup_struct->string_spread+1)*Input-
>sup_struct->string_wt;
    printf("\n\tEnter deck planking dimensions");
    printf("\n\t\theight(inches): ");
    Input->sup_struct->plank_ht = test_input_double(1.0);
    printf("\t\twidth(inches): ");
    Input->sup_struct->plank_w = test_input_double(1.0);
    Input->sup_struct->plank_wt = Input->sup_struct->plank_ht/12*Input->sup_struct->plank_w/12*Input-
>sup_struct->sup_struct_prop->unit_wt;
    get_load_info(Input);
    get_bent_info(Input);
    Input->pier_period = sqrt(Input->spring_tot/Input->weight_tot);
    Input->lat_cap_per_ft = Input->lat_cap/Input->pier_length;
    clean();
    printf("\n\tDo you have physical size data on the ship");
    printf("\n\tyou expect to moor at this pier (y/n)? ");
    choice1 = getche();

    while(choice1 != 'y'&&choice1 != 'Y'&&choice1 != 'n'&&choice1 != 'N')
        {
            prt_inv();
            printf("\n\n\tDo you have measurement data for the ship (y/n)? ");
            choice1 = getche();
        }

    if(choice1 == 'y'||choice1 == 'Y')
        {
            Input->ship = 1;
            get_ship_data(Input);
        }

    if(Input->ship == 1)
        {
            Input->est_ton_ship = -2.97522+3.264*exp((Input->lat_cap_per_ft-302.97)/509.75);

            if(Input->est_ton_ship > 100.0)
                Input->est_ton_ship = 100.0;

            if(Input->est_ton_ship < 0.0)
```

```
                Input->est_ton_ship = 0.0;
        }
}

/*********************************************************/

void get_env_info(PierData *Input)
{
    Env *temp;
    Force *ftemp;
    Wave *wtemp;
    char rpt, bio;
    int flag = 0;
    double w_ang, c_ang, wave_ang, airy_d, airy_h, airy_ha;
    double SK_coef, SD_a, SD_b;

    Input->env_data = calloc(1,sizeof(Env));

    if(Input->env_data == NULL)
        {
            prt_err("Environmental");
        }

    temp = Input->env_data;
    Input->forces = calloc(1,sizeof(Force));

    if(Input->forces == NULL)
        {
            prt_err("Forces");
        }

    ftemp = Input->forces;
    Input->wave_data = calloc(1,sizeof(Wave));

    if(Input->wave_data == NULL)
        {
            prt_err("Wave");
        }

    wtemp = Input->wave_data;
    printf("\n\tWhen measuring wind speed, observations should be of");
    printf("\n\ta sustained wind that lasts for about 30 seconds.");
    printf("\n\n\tPlease enter the wind speed(mph): ");
    temp->wind_spd = test_input_double(0.0);

    if(temp->wind_spd > 0.0)
        {
            printf("\n\tHow high from the water's surface was this");
            printf("\n\tmeasurement taken(feet): ");
            temp->wind_spd_ht = test_input_double(0.0);
            printf("\n\tUsing the pier as the reference datum, you need to input the");
            printf("\n\twind angle. Standing at the foot of the pier, orient a compass");
            printf("\n\tsuch that North is the seaward end of the pier and represents");
```

```
        printf("\n\t-0- or 360 degrees.");
        printf("\n\n\tFrom what angle is the wind coming(deg): ");
        w_ang = test_input_double(0.0);

        if(w_ang > 360.0)
            {
                while(flag == 0)
                    {
                        prt_inv();
                        printf("\n\n\tWhat is the wind angle(deg)?: ");
                        w_ang = test_input_double(0.0);

                        if(w_ang <= 360.0)
                            flag = 1;
                    }
            }

        if(w_ang > 180.0)
            w_ang -= 180.0;
            else
            if(w_ang < 180.0)
                w_ang += 180.0;

        temp->wind_angle_deg = w_ang;
        temp->wind_angle = temp->wind_angle_deg*PI/180;
        temp->wind_10 = exp((log(temp->wind_spd_ht/32.8084))/7)*temp->wind_spd;
        ftemp->wind = 0.00256*pow(temp->wind_10,2);
        ftemp->wind_70 = (0.00256*pow(70,2)*sin(temp->wind_angle));
        ftemp->wind_x = (ftemp->wind*sin(temp->wind_angle));
    }

clean();
printf("\n\tTake current speed obserations at the waters surface.");
printf("\n\tPlease enter the speed of the current(ft per sec): ");
temp->current_spd = test_input_double(0.0);
temp->current_spd = (temp->current_spd*3600/5280);

if(temp->current_spd > 0.0)
    {
        printf("\n\tUse the pier as the reference datum as you did with the");
        printf(" wind.\n\n\tFrom what angle is the current coming(deg): ");
        c_ang = test_input_double(0.0);

        if(c_ang > 360.0)
            {
                while(flag == 0)
                    {
                        prt_inv();
                        printf("\n\n\tWhat is the current angle(deg)?: ");
                        c_ang = test_input_double(0.0);

                        if(c_ang <= 360.0)
                            flag = 1;
```

```
                }
            }

        if(c_ang > 180.0)
            c_ang -= 180.0;
            else
            if(c_ang < 180.0)
                c_ang += 180.0;

        temp->current_angle_deg = c_ang;
        temp->current_angle =   temp->current_angle_deg*PI/180;
    }

clean();
printf("\n\tWave height measurements are from peak to trough and should be");
printf("\n\tof the average wave profile. Measurements should be taken near");
printf("\n\tthe end of the pier or at the area where ship will berth.");
printf("\n\n\tPlease enter the wave height(inches): ");
wtemp->wave_H = test_input_double(0.0)/12;

if(wtemp->wave_H > 0.0)
    {
        printf("\n\tEnter the depth where wave measurements were taken(ft)?: ");
        wtemp->wave_d = test_input_double(1.0);

        if(wtemp->wave_d < 15.0)
            {
                prt_inv();
                printf("\n\n\tIt is HIGHLY unlikely that a ship will be able to");
                printf("\n\tsafely moor in %.1lf ft of water.", wtemp->wave_d);
                printf("\n\n\tYou need to either correct your input or take a");
                printf("\n\tmeasurement at a more appropriate depth...or, we can");
                printf("\n\tgo with your original input...");
                printf("\n\n\tWhat is the depth where wave measurements were taken(ft)?: ");
                wtemp->wave_d = test_input_double(0.0);
            }

        printf("\n\tUse the pier as the reference datum as you");
        printf("\n\tdid with the wind and current.");
        printf("\n\n\tFrom what angle are the waves coming(deg): ");
        wave_ang = test_input_double(0.0);

        if(wave_ang > 360.0)
            {
                while(flag == 0)
                    {
                        prt_inv();
                        printf("\n\n\tWhat is the wave angle(deg)?: ");
                        wave_ang = test_input_double(0.0);

                        if(wave_ang <= 360.0)
                            flag = 1;
                    }
```

```
            }

        if(wave_ang > 180.0)
            wave_ang -= 180.0;
            else
            if(wave_ang < 180.0)
                wave_ang += 180.0;

        wtemp->wave_angle_deg = wave_ang;
        wtemp->wave_angle =   wtemp->wave_angle_deg*PI/180;
        printf("\n\tHow much time passes between successive wave crests(sec): ");
        wtemp->wave_T = test_input_double(0.0);
        wtemp->lambda_0 = (GV*pow(wtemp->wave_T,2))/(2*PI);
        wtemp->wave_c = (4*wtemp->wave_d*pow(PI,2))/(GV*pow(wtemp->wave_T,2));

        if(wtemp->wave_c < 2.0)
            wtemp->wave_k = (sqrt(wtemp->wave_c)*(1+0.169*wtemp->wave_c))/wtemp->wave_d;
            else
            wtemp->wave_k = (wtemp->wave_c*(1+2.0*exp(-wtemp->wave_c)))/wtemp->wave_d;

        wtemp->lambda_A = ((2*PI)/wtemp->wave_k);
        wtemp->wave_umax = (PI*wtemp->wave_H*wtemp->lambda_0)/(wtemp->wave_T*wtemp-
>lambda_A);
        airy_d = wtemp->wave_d/(GV*pow(wtemp->wave_T,2));
        airy_h = wtemp->wave_H/(GV*pow(wtemp->wave_T,2));
        airy_ha = (0.00103-(0.0017/(1+exp((airy_d-0.00549)/0.01306))));

        if(airy_d <= 0.07&&airy_h <= airy_ha)
            wtemp->wave_Airy = 'Y';
        else
            if(airy_d > 0.07&&airy_h <= 0.00103)
                wtemp->wave_Airy = 'Y';
            else
                wtemp->wave_Airy = 'N';

        SK_coef = (2*PI*wtemp->wave_d/wtemp->lambda_A);
        wtemp->wave_Si = (1+((1-cosh(SK_coef))/(SK_coef*sinh(SK_coef))));
        SD_a = (0.5+((1-cosh(2*SK_coef))/(2*SK_coef*sinh(2*SK_coef))));
        SD_b = (1+((2*SK_coef)/(sinh(2*SK_coef))));
        wtemp->wave_SD = 0.5+((1/(4*SD_b))*(SD_a));
    }

clean();
printf("\n\tIs there bio-fouling present(y/n)?: ");
bio = getche();
printf("\n");

while(bio != 'y'&&bio != 'Y'&&bio != 'n'&&bio != 'N')
    {
        prt_inv();
        printf("\n\tIs there bio-fouling(y/n)?: ");
        bio = getche();
        printf("\n");
```

```
        }

    if(bio == 'y'||bio == 'Y')
        {
            printf("\n\n\tHow thick(inches)?: ");
            temp->fouling_factor = test_input_double(0.0);
        }

    clean();
    printf("\n\tWere Rapid Penetration Tests(RPT) performed(y/n)?: ");
    rpt = getche();
    printf("\n");
    flag = 0;

    while(rpt != 'y'&&rpt != 'Y'&&rpt != 'n'&&rpt != 'N')
        {
            prt_inv();
            printf("\n\n\tWere RPTs performed(y/n)?: ");
            rpt = getche();
            printf("\n");
        }

    clean();

    if(rpt == 'y'||rpt == 'Y')
        get_soil_info(temp);

    clean();
}

/*************************************************************/

void get_bent_info(PierData *Input)
{
    Bent *temp;
    int i;
    char batters;

    Input->bents = calloc(1,sizeof(Bent));

    if(Input->bents == NULL)
        {
            prt_err("Bent");
        }

    temp = Input->bents;
    Input->pile_uni_min = 1000;

    for(i = 0; i <= Input->no_of_bents-1; i++)
        {
            clean();
            temp->id_num = i+1;
            temp->load_area = Input->pier_width*Input->bent_spread;
```

```
        if(i == Input->no_of_bents-1)
            temp->load_area /= 2;

        printf("\n\tFor Bent #%d, enter the depth(feet): ",temp->id_num);
        temp->depth = test_input_double(0.0);
        printf("\n\n\tEnter the distance from the water surface to the");

        if(Input->sup_struct->bracing == 'y'||Input->sup_struct->bracing == 'Y')
            printf("\n\tpoint where the pile meets the cross bracing(feet): ");
        else
            printf("\n\tpoint where the pile meets the pile cap(feet): ");

        temp->freeboard = test_input_double(0.0);
        temp->dead_wt += Input->sup_struct->cap_wt;
        temp->dead_wt += Input->sup_struct->string_wt*(Input->pier_width/Input->sup_struct-
>string_spread+1);
        temp->dead_wt += Input->sup_struct->plank_wt*Input->pier_width*Input->bent_spread;
        temp->weight_tot += 1.15*temp->dead_wt;
        temp->dead_wt_per_pile = temp->dead_wt/(Input->piles_per_bent-1);
        temp->pile_length = temp->depth + temp->freeboard;

        if(Input->sup_struct->bracing == 'y'||Input->sup_struct->bracing == 'Y')
            {
                printf("\n\tEnter the distance from where the piling meets the");
                printf("\n\tbracing to the pile cap(feet): ");
                temp->brace_2_cap = test_input_double(0.0);
            }

        temp->pile_length_tot = temp->depth + temp->freeboard + temp->brace_2_cap;

        if(Input->batters->batters == 'y'||Input->batters->batters == 'Y')
            {
                printf("\n\tDoes this bent have batter pilings(y/n)?: ");
                batters = getche();

                while(batters != 'y'&&batters != 'Y'&&batters != 'n'&&batters != 'N')
                    {
                        prt_inv();
                        printf("\n\n\tAre there batter pilings in this bent(y/n)?: ");
                        batters = getche();
                    }

                temp->batters = batters;
            }

        get_pile_info(temp, Input);
        Input->spring_tot += temp->spring_tot;
        Input->weight_tot += temp->weight_tot;
        Input->lat_cap += temp->lat_cap;

        if(i < Input->no_of_bents - 1)
            {
```

```
                temp->next = calloc(1,sizeof(Bent));

                if(temp->next == NULL)
                    {
                        prt_err("Bent");
                    }

                temp = temp->next;
            }
        }
}

/*********************************************************/

void get_soil_info(Env *temp)
{
    int flag = 0;
  char choice;

    while(flag == 0)
        {
            printf("\n\tWhich type of soil did the tests indicate?");
            printf("\n\n\t\t\t1) Soft Silt");
            printf("\n\t\t\t2) Mud");
            printf("\n\t\t\t3) Clay - Very Soft");
            printf("\n\t\t\t4) Clay - Soft");
            printf("\n\t\t\t5) Clay - Medium");
            printf("\n\t\t\t6) Clay - Stiff");
            printf("\n\t\t\t7) Sand - Loose");
            printf("\n\t\t\t8) Sand - Medium");
            printf("\n\t\t\t9) Sand - Dense/Gravel");
            printf("\n\n\t\t\tEnter choice: ");
            choice = getche();

            switch(choice)
                {
                    case '1':
                        strcpy(temp->soil_type, "Soft Silt");
                        temp->soil_flag = 1;
                        flag = 1;
                        break;

                    case '2':
                        strcpy(temp->soil_type, "Mud");
                        temp->soil_flag = 1;
                        flag = 1;
                        break;

                    case '3':
                        strcpy(temp->soil_type, "Very Soft Clay");
                        temp->soil_flag = 1;
                        flag = 1;
                        break;
```

```
                case '4':
                    strcpy(temp->soil_type, "Soft Clay");
                    temp->soil_flag = 1;
                    flag = 1;
                    break;

                case '5':
                    strcpy(temp->soil_type, "Medium Clay");
                    temp->soil_flag = 2;
                    flag = 1;
                    break;

                case '6':
                    strcpy(temp->soil_type, "Stiff Clay");
                    temp->soil_flag = 3;
                    flag = 1;
                    break;

                case '7':
                    strcpy(temp->soil_type, "Loose Sand");
                    temp->soil_flag = 2;
                    flag = 1;
                    break;

                case '8':
                    strcpy(temp->soil_type, "Medium Sand");
                    temp->soil_flag = 3;
                    flag = 1;
                    break;

                case '9':
                    strcpy(temp->soil_type, "Dense Sand & Gravel");
                    temp->soil_flag = 3;
                    flag = 1;
                    break;

                default:
                    prt_inv();
                    break;
            }
        }
}

/*********************************************************/

void get_pile_info(Bent *bent, PierData *Input)
{
    Pile *temp, *prev_temp;
    int i;
    char side[6];

    bent->pile = calloc(1,sizeof(Pile));
```

```
    if(bent->pile == NULL)
        {
            prt_err("Pile");
        }

    temp = bent->pile;

    for(i = 0; i <= Input->piles_per_bent - 1; i++)
        {
            clean();
            temp->id_let[0] = ' ';
            temp->id_let[1] = (char)(i+65);
            temp->diameter = Input->pile_size;
            temp->unsup_length = bent->pile_length+Input->fixity;
            temp->pile_weight = (PI*pow(temp->diameter/12,2)/4)*(bent->pile_length_tot*Input->pile_prop-
>unit_wt-bent->depth*H2O_RG);
            bent->weight_tot += temp->pile_weight;
            temp->length_mud_to_cap = bent->pile_length_tot;
            if(i == 0 || i == Input->piles_per_bent - 1)
                {
                    temp->dead_load = bent->dead_wt_per_pile/2 + 0.075*bent->dead_wt_per_pile;
                    temp->load_area = bent->load_area/(Input->piles_per_bent - 1)/2;
                }
            else
                {
                    temp->dead_load = bent->dead_wt_per_pile;
                    temp->load_area = bent->load_area/(Input->piles_per_bent - 1);
                }

            if(Input->sup_struct->bracing == 'y'||Input->sup_struct->bracing == 'Y')
                temp->eff_length = 0.5*temp->unsup_length;
            else
                temp->eff_length = 2*temp->unsup_length;

            printf("\n\tFor Bent #%d, ", bent->id_num);
            printf("pile: %s \n\tenter the following:", temp->id_let);
            get_cond(bent, temp, Input, i+1);

            if(i < Input->piles_per_bent-1)
                {
                    temp->next = calloc(1,sizeof(Pile));

                    if(temp->next == NULL)
                        {
                            prt_err("Pile");
                        }
                    prev_temp = temp;
                    temp = temp->next;
        temp->prev = prev_temp;
                }
        }
```

```
    if(bent->batters == 'y'||bent->batters == 'Y')
        {
            for(i = 0; i <= 1; i++)
                {
                    temp->next = calloc(1,sizeof(Pile));

                    if(temp->next == NULL)
                        {
                            prt_err("Batter Pile");
                        }

                    temp = temp->next;

                    if(i == 0)
                        {
                            strcpy(temp->id_let, "bL");
                            strcpy(side, "left");
                        }
                    else
                        {
                            strcpy(temp->id_let, "bR");
                            strcpy(side, "right");
                        }

                    temp->unsup_length = (bent->pile_length + Input->fixity)/(sin(Input->batters->batter_angle));
                    temp->diameter = Input->batters->batter_size;
                    temp->eff_length = 0.7*temp->unsup_length;
                    temp->pile_weight = (PI*pow(temp->diameter/12,2)/4)*(bent->pile_length_tot*Input-
>pile_prop->unit_wt - bent->depth*H2O_RG)/(sin(Input->batters->batter_angle));
                    bent->weight_tot += temp->pile_weight;
                    clean();
                    printf("\n\tFor Bent #%d, ", bent->id_num);
                    printf("the %s batter pile, \n\tenter the following:", side);
                    get_cond(bent, temp, Input, 0);
                }
        }
}

/*********************************************************/

void get_prop_info(Props *matl, int pflag)
{
    char choice;
    int flag = 0;

    while(flag == 0)
        {
            printf("\n\n\tWhen identifying the type of wood, if the type you are looking");
            printf("\n\tfor is not present, choose the closest or next weaker type. ");
            printf("\n\n\t\t\t1) Douglas Fir");
            printf("\n\t\t\t2) Southern Pine");
            printf("\n\t\t\t3) White Oak");
            printf("\n\t\t\t4) Red Oak");
```

```
printf("\n\t\t\t5) Northern Red Oak");
printf("\n\t\t\t6) Mixed Oak");
printf("\n\t\t\t7) Mixed Hardwoods");
printf("\n\t\t\t8) Greenheart - usually only piles");
printf("\n\t\t\t9) Ekki (Azobe) - usually only piles");
printf("\n\n\t\t\tEnter choice: ");
choice = getche();

switch(choice)
   {
      case '1':
         if(pflag == 1)
            {
               matl->bending_mod = 1700000.0;
               matl->sig_pub_bend = 1750.0;
            }
         else
            {
               matl->bending_mod = 1800000.0;
               matl->sig_pub_bend = 1900.0;
            }

         matl->sig_pub_comp = 1350.0;
         matl->unit_wt = 34.0;
         strcpy(matl->tim_type, "Douglas Fir");
         flag = 1;
         break;

      case '2':
         if(pflag == 1)
            {
               matl->bending_mod = 1600000.0;
               matl->sig_pub_bend = 1500.0;
            }
         else
            {
               matl->bending_mod = 1600000.0;
               matl->sig_pub_bend = 1800.0;
            }

         matl->sig_pub_comp = 975.0;
         matl->unit_wt = 36.0;
         strcpy(matl->tim_type,"Southern Pine");
         flag = 1;
         break;

      case '3':
         if(pflag == 1)
            {
               matl->bending_mod = 1000000.0;
               matl->sig_pub_bend = 1300.0;
            }
         else
```

```
        {
            matl->bending_mod = 1000000.0;
            matl->sig_pub_bend = 1400.0;
        }

    matl->sig_pub_comp = 850.0;
    matl->unit_wt = 48.0;
    strcpy(matl->tim_type,"White Oak");
    flag = 1;
    break;

case '4':
    if(pflag == 1)
        {
            matl->bending_mod = 1200000.0;
            matl->sig_pub_bend = 1250.0;
        }
    else
        {
            matl->bending_mod = 1200000.0;
            matl->sig_pub_bend = 1350.0;
        }

    matl->sig_pub_comp = 875.0;
    matl->unit_wt = 44.0;
    strcpy(matl->tim_type,"Red Oak");
    flag = 1;
    break;

case '5':
    if(pflag == 1)
        {
            matl->bending_mod = 1300000.0;
            matl->sig_pub_bend = 1500.0;
        }
    else
        {
            matl->bending_mod = 1300000.0;
            matl->sig_pub_bend = 1600.0;
        }

    matl->sig_pub_comp = 1000.0;
    matl->unit_wt = 44.0;
    strcpy(matl->tim_type,"Northern Red Oak");
    flag = 1;
    break;

case '6':
    if(pflag == 1)
        {
            matl->bending_mod = 1000000.0;
            matl->sig_pub_bend = 1250.0;
        }
```

```
            else
                {
                    matl->bending_mod = 1000000.0;
                    matl->sig_pub_bend = 1350.0;
                }

            matl->sig_pub_comp = 875.0;
            matl->unit_wt = 46.0;
            strcpy(matl->tim_type,"Mixed Oak");
            flag = 1;
            break;

        case '7':
            if(pflag == 1)
                {
                    matl->bending_mod = 1500000.0;
                    matl->sig_pub_bend = 1550.0;
                }
            else
                {
                    matl->bending_mod = 1500000.0;
                    matl->sig_pub_bend = 1650.0;
                }

            matl->sig_pub_comp = 875.0;
            matl->unit_wt = 42.0;
            strcpy(matl->tim_type,"Mixed Hardwoods");
            flag = 1;
            break;

        case '8':
            matl->bending_mod = 3700000.0;
            matl->sig_pub_comp = 3400.0;
            matl->sig_pub_bend = 4500.0;
            matl->unit_wt = 66.0;
            strcpy(matl->tim_type,"Greenheart");
            flag = 1;
            break;

        case '9':
            matl->bending_mod = 3000000.0;
            matl->sig_pub_comp = 3400.0;
            matl->sig_pub_bend = 4500.0;
            matl->unit_wt = 65.5;
            strcpy(matl->tim_type,"Ekki");
            flag = 1;
            break;

        default:
            prt_inv();
            break;
    }
}
```

```
}

/*********************************************************/

double test_input(int num, double check)
{
    char in_value[20] = {0}, choice;

    double out_value = 0.0;
    int j, i, flag = 0, ctr = 0;

    while(flag == 0)
        {
            scanf("%s", in_value);

            for(i = 0; i < strlen(in_value); i++)
                {
                    if(isdigit(in_value[i]) == 0)
                        flag = 1;
                }

            if(strlen(in_value) > num||flag == 1)
                {
                    ctr++;

                    if(ctr >= 3)
                        {
                            printf("\n\tThat's 3 incorrect inputs in a row.");
                            printf("\n\tDo you wish to quit (y/n)? ");
                            choice = getche();

                            if(quit(choice))
                                {
                                    printf("\n\n\tLater!!!");
                                    exit(0);
                                }
                            else
                                {
                                    ctr = 0;
                                }
                        }
                    prt_inv();
                    printf("\n\tRe-enter value: ");
                    flag = 0;
                    in_value[0] = '\0';
                }
            else
                flag = 2;
        }

    for(i = 0, j = strlen(in_value); j > 0; j--,i++)
        out_value += (double)(((int)in_value[i]-48)*pow(10, j-1));
```

```
    if(out_value < check)
        {
            prt_inv();
            printf("\n\tRe-enter value: ");
            out_value = test_input(num, check);
        }

    return(out_value);
}

/*************************************************************/

int quit(char choice)
{
    char temp = 'n';

    if(choice == 'y'||choice == 'Y')
        return(1);
    else
        {
            if(choice == 'n'||choice == 'N')
                return(0);
            else
                {
                    printf("Invalid selection. Quit(y/n)? ");
                    temp = getche();
                    return(quit(temp));
                }
        }
}

/*************************************************************/

void get_cond(Bent *bent, Pile *temp, PierData *Input, int pile_no)
{
    Force *ftemp;
    int flag = 0, flag1 = 0;
    double sigma = 0.0, Sigma_calc = 0.0, I = 0.0, b = 0.0, h = 0.0, S = 0.0;
    double diam = 0.0, eff_l = temp->eff_length, A = 0.0, x = 0.0, y = 0.0;
    double bat_temp = 0.0, wc_moment = 0.0;
    char choice, choice1;

    ftemp = Input->forces;

    while(flag == 0)
        {
            printf("\n\n\tCondition code for this pile:");
            printf("\n\n\t\t1) ND   2) MN   3) MD   4) MJ   5) SV");
            printf("\n\n\t\t\tEnter choice: ");
            choice = getche();

            switch(choice)
                {
```

```
case '1':
    strcpy(temp->condition, "ND");
    diam = temp->diameter;
    I = (PI * pow(diam/2, 4))/4;
    A = (PI * pow(diam/2, 2));
    flag = 1;

    if(Input->sup_struct->bracing == 'y'||Input->sup_struct->bracing == 'Y')
        flag1 = 1;
    else
        flag1 = 2;

    temp->flag = 0;
    break;

case '2':
    strcpy(temp->condition, "MN");
    printf("\n\tEnter the reduced piling diameter size(inches): ");
    temp->diameter = test_input(2, 1.0);
    diam = temp->diameter;
    I = (PI * pow(diam/2, 4))/4;
    A = (PI * pow(diam/2, 2));
    flag = 1;

    if(Input->sup_struct->bracing == 'y'||Input->sup_struct->bracing == 'Y')
        flag1 = 1;
    else
        flag1 = 2;

    temp->flag = 0;
    break;

case '3':
    strcpy(temp->condition, "MD");
    while(flag1 == 0)
        {
            printf("\n\n\tIs the defective portion of this pile");
            printf(" (c)ircular\n\t or (r)ectangular (\"c\" or \"r\")? ");
            choice1 = getche();
            printf("\n");
            switch(choice1)
                {
                    case 'c':
                    case 'C':
                        printf("\n\tEnter the reduced piling diameter size(inches): ");
                        temp->diameter = test_input_double(1.0);
                        diam = temp->diameter;
                        A = (PI * pow(diam/2, 2));
                        I = (PI * pow(diam/2, 4))/4;

                        if(Input->sup_struct->bracing == 'y'||Input->sup_struct->bracing == 'Y')
                            flag1 = 1;
                        else
```

```
                    flag1 = 2;

                break;

            case 'r':
            case 'R':
                printf("\n\tSuperimpose a rectangle on the section. ");
                printf("Enter the\n\tfollowing dimensions -");
                printf("\n\t\tWidth(inches): ");
                h = test_input_double(0.0);
                printf("\t\tLength(inches): ");
                b = test_input_double(0.0);
                A = (h * b);
                I = (b*(pow(h, 3))/12);
                flag1 = 3;
                break;

            default:
                prt_inv();
                break;
        }
    }

flag = 1;
temp->flag = 0;
break;

case '4':
    strcpy(temp->condition, "MJ");
    while(flag1 == 0)
        {
            printf("\n\n\tIs the defective portion of this pile");
            printf(" (c)ircular, (r)ectangular");
            printf("or (i)ncapable of supporting a load (\"c\", \"r\" or \"i\")? ");
            choice1 = getche();
            printf("\n");
            switch(choice1)
                {
                    case 'c':
                    case 'C':
                        printf("\n\tEnter the reduced piling diameter size(inches): ");
                        temp->diameter = test_input_double(1.0);
                        diam = temp->diameter;
                        A = (PI * pow(diam/2, 2));
                        I = (PI * pow(diam/2, 4))/4;

                        if(Input->sup_struct->bracing == 'y'||Input->sup_struct->bracing == 'Y')
                            flag1 = 1;
                        else
                            flag1 = 2;

                        temp->flag = 0;
                        break;
```

```
                        case 'r':
                        case 'R':
                            printf("\n\tSuperimpose a rectangle on the section. ");
                            printf("Enter the\n\tfollowing dimensions -");
                            printf("\n\t\tWidth(inches): ");
                            h = test_input_double(0.0);
                            printf("\t\tLength(inches): ");
                            b = test_input_double(0.0);
                            A = (h * b);
                            I = (b*(pow(h, 3))/12);

                            flag1 = 3;
                            temp->flag = 0;
                            break;

                        case 'i':
                        case 'I':
                            I = 0.0;
                            A = 0.0;

                            flag1 = 4;
                            temp->flag = 1;
                            break;

                        default:
                            prt_inv();
                            break;
                    }
                }

            flag = 1;
            break;

        case '5':
            strcpy(temp->condition, "SV");
            I = 0.0;
            A = 0.0;

            flag = 1;
            flag1 = 4;
            temp->flag = 1;
            break;

        default:
            prt_inv();
            break;
    }
}

x = pow(PI,2)*I*Input->pile_prop->bending_mod;
y = A*pow(eff_l*12,2)*Input->factor_of_safety;
```

```
    if(A == 0.0)
        Sigma_calc = 0.0;
    else
        Sigma_calc = x/y;

    if(Sigma_calc < Input->pile_prop->sig_pub_comp)
        sigma = Sigma_calc;
    else
        sigma = Input->pile_prop->sig_pub_comp;

    if(temp->id_let[0] == 'b')
        {
            temp->vert_load = A*sigma;

            if(temp->vert_load < 0)
                temp->vert_load = 0;

            bat_temp = temp->vert_load*cos(Input->batters->batter_angle);

            if(temp->id_let[1] == 'R')
                {
                    temp->lat_load = 1.333*bat_temp;
                    bent->lat_cap += temp->lat_load;
                }
        }
    else
        {
            if(temp->flag == 1)
                {
                    if(pile_no == 1||pile_no == Input->piles_per_bent)
                        {
                            temp->vert_load = Input->loads->pile_pt_miss_end - temp->dead_load;
                            if(pile_min_p < temp->vert_load)
                                temp->vert_load = pile_min_p;
                            temp->uni_load_psi = Input->loads->pile_uni_miss_end - temp->dead_load/temp-
>load_area/144;
                            if(pile_min_u < temp->uni_load_psi)
                                temp->uni_load_psi = pile_min_u;
                        }
                    else
                        {
                            temp->vert_load = Input->loads->pile_pt_miss - temp->dead_load;
                            if(pile_min_p < temp->vert_load)
                                temp->vert_load = pile_min_p;
                            temp->uni_load_psi = Input->loads->pile_uni_miss - temp->dead_load/temp-
>load_area/144;
                            if(pile_min_u < temp->uni_load_psi)
                                temp->uni_load_psi = pile_min_u;
                        }
                }
            else
                {
                    temp->vert_load = A*sigma - temp->dead_load;
```

```
                temp->uni_load_psi = temp->vert_load/temp->load_area/144;

                if(temp->vert_load <= 0)
                    {
                        if(pile_no == 1||pile_no == Input->piles_per_bent)
                            {
                                temp->vert_load = Input->loads->pile_pt_miss_end - temp->dead_load;
                                if(pile_min_p < temp->vert_load)
                                    temp->vert_load = pile_min_p;
                                if(temp->vert_load <= 0)
                                    temp->vert_load = 0;
                            }
                        else
                            {
                                temp->vert_load = Input->loads->pile_pt_miss - temp->dead_load;
                                if(pile_min_p < temp->vert_load)
                                    temp->vert_load = pile_min_p;
                                if(temp->vert_load <= 0)
                                    temp->vert_load = 0;
                            }
                    }
                else
                    if((Input->loads->pile_pt_miss - temp->dead_load) < pile_min_p)
                        {
                            if(temp->vert_load < (Input->loads->pile_pt_miss - temp->dead_load))
                                temp->vert_load = (Input->loads->pile_pt_miss - temp->dead_load);
                        }
                    else
                            temp->vert_load = pile_min_p;

                if(temp->uni_load_psi <= 0)
                    {
                        if(pile_no == 1||pile_no == Input->piles_per_bent)
                            {
                                temp->uni_load_psi = Input->loads->pile_uni_miss_end - (temp-
>dead_load/temp->load_area/144);
                                if(pile_min_u < temp->uni_load_psi)
                                    temp->uni_load_psi = pile_min_u;
                                if(temp->uni_load_psi <= 0)
                                    temp->uni_load_psi = 0;
                            }
                        else
                            {
                                temp->uni_load_psi = Input->loads->pile_uni_miss - temp->dead_load/temp-
>load_area/144;
                                if(pile_min_u < temp->uni_load_psi)
                                    temp->uni_load_psi = pile_min_p;
                                if(temp->uni_load_psi <= 0)
                                    temp->uni_load_psi = 0;
                            }
                    }
                else
                    if((Input->loads->pile_uni_miss - temp->dead_load/temp->load_area/144) < pile_min_u)
```

```
                        {
                            if(temp->uni_load_psi < (Input->loads->pile_uni_miss - temp->dead_load/temp-
>load_area/144))
                                temp->uni_load_psi = (Input->loads->pile_uni_miss - temp->dead_load/temp-
>load_area/144);
                        }
                    else
                            temp->uni_load_psi = pile_min_u;
                }
            if(Input->pile_uni_min > temp->uni_load_psi)
                {
                    Input->pile_uni_min = temp->uni_load_psi;
                    Input->loads->pile_uni_min = Input->pile_uni_min;
                }

            wc_moment = (ftemp->wave_M_x_d*bent->depth*12)+(ftemp->current_pier_M_x*pow(bent-
>depth*12,2));
            temp->spring_cap = 3*Input->pile_prop->bending_mod*I/pow(eff_l*12,3);
            bent->spring_tot += temp->spring_cap;
            S = Input->pile_prop->sig_pub_bend;

            if(flag1 == 1)
                {
                    temp->lat_load = 2*(S*PI*pow(diam/2,3)/4 - wc_moment)/(eff_l*12);
                    bent->lat_cap += temp->lat_load;
                }

            if(flag1 == 2)
                {
                    temp->lat_load = (S*PI*pow(diam/2,3)/4 - wc_moment)/(eff_l*12);
                    bent->lat_cap += temp->lat_load;
                }

            if(flag1 == 3)
                {
                    temp->lat_load = 2*(S*b*pow(h,2)/4 - wc_moment)/(eff_l*12);
                    bent->lat_cap += temp->lat_load;
                }

            if(flag1 == 4)
                {
                    temp->lat_load = 0.0;
                    bent->lat_cap += temp->lat_load;
                }
        }
}

/*********************************************************/

void get_load_info(PierData *Input)
{
    Load *temp;
    double hs_a = 0.0, hs_b = 0.0, h_a = 0.0, h_b = 0.0, pu_a = 0.0, pu_b = 0.0;
```

```
    double cu_a = 0.0, cu_b = 0.0, su_a = 0.0, su_b = 0.0, sp_a = 0.0, sp_b = 0.0;
    double ca_a = 0.0, ca_b = 0.0, cp_a = 0.0, cp_b = 0.0, sig_sup = 0.0;
    double sig_adj_pl = 0.0;

    Input->loads = calloc(1,sizeof(Load));

    if(Input->loads == NULL)
        {
            prt_err("Load");
        }

    temp = Input->loads;

    if(Input->sup_struct->plank_w >= 5.0)
        {
            if(Input->sup_struct->plank_ht <= 2.0)
                sig_adj_pl = 1.22;
            else
                if(Input->sup_struct->plank_ht <= 3.0)
                    sig_adj_pl = 1.16;
                else
                    sig_adj_pl = 1.11;
        }
    else
        if(Input->sup_struct->plank_ht <= 2.0)
            sig_adj_pl = 1.10;
        else
            if(Input->sup_struct->plank_ht <= 3.0)
                sig_adj_pl = 1.04;
            else
                sig_adj_pl = 1.00;

    sig_sup = Input->sup_struct->sup_struct_prop->sig_pub_bend;

    if(Input->bent_spread <= 23.9)
        {
            hs_a = (4/Input->sup_struct->string_spread)*2*Input->sup_struct->string_w*pow(Input->sup_struct-
>string_ht,2)*sig_sup;
            hs_b = 3*Input->bent_spread*12;
            temp->hs_limit_wheel_str = hs_a/hs_b;
        }
    else
        {
            hs_a = (4/Input->sup_struct->string_spread)*Input->sup_struct->string_w*pow(Input->sup_struct-
>string_ht,2)*sig_sup;
            hs_b = 3*pow((Input->bent_spread-7)*12,2);
            temp->hs_limit_wheel_str = Input->bent_spread*12*hs_a/hs_b;
        }

    if(Input->bent_spread <= 27.0)
        {
            h_a = (4/Input->sup_struct->string_spread)*2*Input->sup_struct->string_w*pow(Input->sup_struct-
>string_ht,2)*sig_sup;
```

```
            h_b = 3*Input->bent_spread*12;
            temp->h_limit_wheel_str = h_a/h_b;
        }
    else
        {
            h_a = (4/Input->sup_struct->string_spread)*Input->sup_struct->string_w*pow(Input->sup_struct-
>string_ht,2)*sig_sup;
            h_b = (1.862*Input->bent_spread-9.606)*12;
            temp->h_limit_wheel_str = h_a/h_b;
        }

    sp_a = (4/Input->sup_struct->string_spread)*4*Input->sup_struct->string_w*pow(Input->sup_struct-
>string_ht,2)*sig_sup;
    sp_b = 6*Input->bent_spread*12;
    temp->string_pt_lim = sp_a/sp_b;

    su_a = (4/Input->sup_struct->string_spread)*8*Input->sup_struct->string_w*pow(Input->sup_struct-
>string_ht,2)*sig_sup;
    su_b = 6*pow(Input->bent_spread*12,2);
    temp->string_uni_lim_psi = su_a/su_b/(Input->sup_struct->string_spread*12);

    ca_a = Input->sup_struct->cap_w*pow(Input->sup_struct->cap_ht,2)*sig_sup;
    ca_b = 72*(-124.02+92.39*(1-exp(-Input->pile_spread/13.59))+169.237*(1-exp(-Input-
>pile_spread/3.203)))/100;
    temp->axle_lim_cap = ca_a/ca_b;

    cp_a = Input->sup_struct->cap_w*pow(Input->sup_struct->cap_ht,2)*sig_sup;
    cp_b = 6*0.07922555*Input->pile_spread*12;
    temp->cap_pt_lim = cp_a/cp_b;

    cu_a = Input->sup_struct->cap_w*pow(Input->sup_struct->cap_ht,2)*sig_sup;
    cu_b = 6*0.1056338*Input->bent_spread*12*Input->pile_spread*12;
    temp->cap_uni_lim_psi = cu_a/cu_b;

    pu_a = pow(Input->sup_struct->plank_ht,2)*sig_sup*sig_adj_pl;
    pu_b = 6*0.1056338*Input->sup_struct->string_spread*12;
    temp->plank_uni_lim_psi = pu_a/pu_b;

    temp->pile_uni_min = Input->pile_uni_min;
    temp->pile_pt_miss = (Input->sup_struct->cap_w*pow(Input->sup_struct-
>cap_ht,2)*sig_sup)/(6*0.261686*Input->pile_spread*12);
    temp->pile_uni_miss = (Input->sup_struct->cap_w*pow(Input->sup_struct-
>cap_ht,2)*sig_sup)/(6*0.1996*Input->pile_spread*12*Input->bent_spread*12);
    temp->pile_pt_miss_end = (Input->sup_struct->cap_w*pow(Input->sup_struct-
>cap_ht,2)*sig_sup)/(6*Input->pile_spread*12);
    temp->pile_uni_miss_end = (2*Input->sup_struct->cap_w*pow(Input->sup_struct-
>cap_ht,2)*sig_sup)/(6*pow(Input->pile_spread*12,2)*Input->bent_spread*12);

    temp->fork_lim_5 = check_fork(6.25, 10000, Input);
    temp->fork_lim_6 = check_fork(6.25, 11500, Input);
    temp->fork_lim_75 = check_fork(6.333, 14500, Input);
    temp->fork_lim_8 = check_fork(6.333, 15250, Input);
    temp->fork_lim_10 = check_fork(6.333, 17500, Input);
```

```
    temp->fork_lim_12= check_fork(6.333, 22150, Input);
    temp->fork_lim_15 = check_fork(6.5, 29000, Input);
    temp->fork_lim_20 = check_fork(8.0, 490100, Input);
}

/*********************************************************/

char check_fork(double a, double lim, PierData *Input)
{
    double p, p_a, p_b;
    double sig_sup = Input->sup_struct->sup_struct_prop->sig_pub_bend;

    if(Input->bent_spread <= (1.707*a))
        {
            p_a = (4/Input->sup_struct->string_spread)*4*Input->sup_struct->string_w*pow(Input->sup_struct-
>string_ht,2)*sig_sup;
            p_b = 6*Input->bent_spread*12;
            p = p_a/p_b;

            if(p >= lim)
                return('Y');
            else
                return('N');
        }
    else
        {
            p_a = (4/Input->sup_struct->string_spread)*2*Input->sup_struct->string_w*pow(Input->sup_struct-
>string_ht,2)*sig_sup;
            p_b = 6*pow((Input->bent_spread-a/2)*12,2);
            p = Input->bent_spread*12*p_a/p_b;

            if(p >= lim)
                return('Y');
            else
                return('N');
        }
}

/*********************************************************/
void get_ship_data(PierData *Input)
{
    Ship *temp;
    double w_load, c_load;
    int k = 1.3;

    Input->ship_data = calloc(1,sizeof(Ship));

    if(Input->ship_data == NULL)
        {
            prt_err("Ship");
        }

    temp = Input->ship_data;
```

```
    printf("\n\n\t\tEnter ship length(feet): ");
    temp->length = test_input_double(1.0);

    if(temp->length <= 80)
        {
            temp->C1 = 1.0;
            temp->C2 = 1.45;
        }
    else
        if(temp->length <= 250)
            {
                temp->C1 = 0.8;
                temp->C2 = 1.37;
            }
    else
        if(temp->length < 655)
            {
                temp->C1 = 0.65;
                temp->C2 = 1.31;
            }
    else
            {
                temp->C1 = 0.5;
                temp->C2 = 1.25;
            }

    printf("\n\t\tEnter ship draft(feet): ");
    temp->draft = test_input_double(1.0);
    printf("\n\t\tFor this next amount, you will need to \"flatten\" the");
    printf("\n\t\tsuperstructure so that the ship is a rectangle sitting on");
    printf("\n\t\tthe water.");
    printf("\n\n\t\tEnter the \"height\" of the rectangle (feet): ");
    temp->freeboard = test_input_double(1.0);
    w_load = k*temp->C1*temp->C2*temp->length*temp->freeboard*Input->forces->wind_x;
    c_load = temp->length*temp->draft*Input->forces->current_ship_x;
    temp->force_tot = w_load + c_load;

    if((Input->lat_cap-fabs(temp->force_tot)) > 0)
        Input->ship = 2;
}

/*********************************************************/
char prn_to_file(PierData *p_info, Flagz *flagz)
{
    FILE *ofp, *rfp;
    Bent *b_temp;
    Pile *p_temp;
    char choice, f_temp, fname[13] = "pier_in0.dat";
    int flag1 = 1;
    double uni_min, p_load, w_load, t_angle, p_min;

    if(flagz->t_flag == 1)
            fname[7] = 't';
```

```
else
   {
      remove("pier_int.dat");
      printf("\n\n\tEnter a number between 0-9. This number will be used");
      printf("\n\twhen saving the user data file so that you can tell the");
      printf("\n\tdifference between input files. For example, if you");
      printf("\n\tinput the number 3, your data file will be saved as:");
      printf("\n\n\t\t\tpier_in3.dat");
      printf("\n\n\tIf you enter a number that has been used previously,");
      printf("\n\tyou will overwrite that file.");
      printf("\n\n\tPlease enter number: ");
      f_temp = (int)(check_num()+48);
      fname[7] = f_temp;

      while(flag1 == 1)
         {
            rfp = fopen(fname, "r");

            if(rfp != NULL)
               {
                  printf("\n\n\tWarning!!! This file already exists.");
                  printf("\n\tDo you wish to overwrite this file(y/n)? ");
                  choice = getche();

                  while(choice != 'y'&&choice != 'Y'&&choice != 'n'&&choice != 'N')
                     {
                        prt_inv();
                        printf("\n\tDo you wish to overwrite this file(y/n)?  ");
                        choice = getche();
                     }

                  if(choice == 'n'||choice == 'N')
                     {
                        printf("\n\n\tPlease enter new number: ");
                        f_temp = (int)(check_num()+48);
                        fname[7] = f_temp;
                        rfp = fopen(fname, "r");
                     }
                  else
                     flag1 = 0;
               }
            else
               flag1 = 0;
         }
      fclose(rfp);
   }

clrscr();

if(p_info->no_of_bents >= 3&&p_info->piles_per_bent >=3&&flagz->t_flag != 1)
   {
      flagz->s_flag = 1;
      pile_plot(p_info, f_temp);
```

```
    }

  ofp = fopen(fname, "w");

  if(ofp == NULL)
    {
      printf("\nERROR: Cannot open user output file!\n");
      exit(-1);
    }

  fprintf(ofp,"\tFile generated: ");
  date_plot(ofp);
  fprintf(ofp,"\n\tThe following information was input in to RSAP.");
  fprintf(ofp,"\n\t(C)1999 - All rights reserved. R. J. Keiter");
  fprintf(ofp,"\n\t=========================================================");

  if(p_info->env_data->wind_spd > 0)
    {
      fprintf(ofp,"\n\n\tThe wind speed was measured at %.0lf mph,",p_info->env_data->wind_spd);
      fprintf(ofp," %.0lf' above \n\t   the water's surface",p_info->env_data->wind_spd_ht);

      if(p_info->env_data->wind_angle_deg > 180)
        t_angle = p_info->env_data->wind_angle_deg - 180;
      else
        t_angle = p_info->env_data->wind_angle_deg + 180;

      fprintf(ofp," at an angle of %.0lf degrees.",t_angle);
    }
  else
    fprintf(ofp,"\n\tThere were no wind data provided.");

  if(p_info->env_data->current_spd > 0)
    {
      fprintf(ofp,"\n\tThe current speed was measured at %.2lf mph",p_info->env_data->current_spd);

      if(p_info->env_data->current_angle_deg > 180)
        t_angle = p_info->env_data->current_angle_deg - 180;
      else
        t_angle = p_info->env_data->current_angle_deg + 180;

      fprintf(ofp," at an \n\t   angle of %.0lf degrees.",t_angle);
    }
  else
    fprintf(ofp,"\n\tThere were no current data provided.");

  if(p_info->wave_data->wave_H > 0)
    {
      fprintf(ofp,"\n\t%.2lf\' waves at %.0lf seconds were observed",p_info->wave_data->wave_H, p_info-
>wave_data->wave_T);

      if(p_info->wave_data->wave_angle_deg > 180)
        t_angle = p_info->wave_data->wave_angle_deg - 180;
      else
```

```
            t_angle = p_info->wave_data->wave_angle_deg + 180;

        fprintf(ofp," in %.0lf' of water at an \n\t    angle of %.0lf degrees.",p_info->wave_data->wave_d,t_angle);

        if(p_info->wave_data->wave_Airy == 'N')
            {
                fprintf(ofp,"\n\n\t*************************************************************");
                fprintf(ofp,"\n\t**************** ATTENTION!!! **********************");
                fprintf(ofp,"\n\t*************************************************************");
                fprintf(ofp,"\n\n\tThese waves do not fall under the theory used here to");
                fprintf(ofp,"\n\tanalyze the forces due to waves. The actual forces will");
                fprintf(ofp,"\n\tmost likely be larger than that calculated and used in ");
                fprintf(ofp,"\n\tthis assessment. Proceed with caution!");
                fprintf(ofp,"\n\n\t*************************************************************");
                fprintf(ofp,"\n\t**************** ATTENTION!!! **********************");
                fprintf(ofp,"\n\t*************************************************************");
            }
    }
  else
    fprintf(ofp,"\n\tThere were no wave data provided.");

  if(p_info->env_data->soil_flag > 0)
    fprintf(ofp,"\n\n\tSoil Type: %s",p_info->env_data->soil_type);
  else
    fprintf(ofp,"\n\n\tNo Soil information provided.");

  fprintf(ofp,"\n\n\tThis analysis was performed using a factor of safety of: ");
  fprintf(ofp,"%.2lf",p_info->factor_of_safety);
  fprintf(ofp,"\n\n\tNumber of bents: %d", p_info->no_of_bents);
  fprintf(ofp,"\n\tThe distance between bents is: %.2lf", p_info->bent_spread);
  fprintf(ofp,"\n\tThe pilings are %.0lf\" %s", p_info->pile_size, p_info->pile_prop->tim_type);

  if(p_info->env_data->fouling_factor > 0)
    fprintf(ofp," with %.0lf\" of bio-fouling.", p_info->env_data->fouling_factor);
  else
    fprintf(ofp,".");

  fprintf(ofp,"\n\tNumber of piles per bent: %d", p_info->piles_per_bent);
  fprintf(ofp,"\n\tDistance between pilings: %.2lf", p_info->pile_spread);

  if(p_info->batters->batters == 'y'||p_info->batters->batters == 'Y')
    {
      fprintf(ofp,"\n\tThis pier has %.0lf\" batter pilings", p_info->batters->batter_size);
      fprintf(ofp," at a %.0lf degree angle.", p_info->batters->batter_angle_deg);
    }

  fprintf(ofp,"\n\n\tThe superstructure is constructed of %s,", p_info->sup_struct->sup_struct_prop->tim_type);

  if(p_info->sup_struct->bracing == 'y'||p_info->sup_struct->bracing == 'Y')
    fprintf(ofp,"\n\t   has cross bracing, ");
  else
    fprintf(ofp,"\n\t   has NO cross bracing, ");
```

```
    if(p_info->sup_struct->cap_type == 's'||p_info->sup_struct->cap_type == 'S')
        fprintf(ofp,"and has a solid type pile cap.");
    else
        fprintf(ofp,"and has a split type pile cap.");

    fprintf(ofp,"\n\tThe pile cap member dimensions are");
    fprintf(ofp," %.0lf\" x %.0lf\".", p_info->sup_struct->cap_w, p_info->sup_struct->cap_ht);
    fprintf(ofp,"\n\tThe stringers are %.0lf\" x %.0lf\"", p_info->sup_struct->string_w, p_info->sup_struct-
>string_ht);
    fprintf(ofp," and are centered %.2lf apart.", p_info->sup_struct->string_spread);
    fprintf(ofp,"\n\tThe deck planking is %.0lf\" x %.0lf\".", p_info->sup_struct->plank_w, p_info->sup_struct-
>plank_ht);
    fprintf(ofp,"\n\n\tThe piling inspection information is as follows:");
    fprintf(ofp,"\n\t   (The number following the bent number is the");
    fprintf(ofp,"\n\t    distance from the bottom to where the piling");
    fprintf(ofp,"\n\t    meets the");

    if(p_info->sup_struct->bracing == 'y'||p_info->sup_struct->bracing == 'Y')
        fprintf(ofp," cross bracing.)");
    else
        fprintf(ofp," pile cap.)");

    b_temp = p_info->bents;

    while(b_temp != NULL)
        {
            fprintf(ofp,"\n\n\tBent #%d - %.0lf'\n\t", b_temp->id_num, b_temp->pile_length);
            p_temp = b_temp->pile;

            while(p_temp != NULL)
                {
                    fprintf(ofp,"  %s: %s", p_temp->id_let, p_temp->condition);
                    p_temp = p_temp->next;
                }
            b_temp = b_temp->next;
        }

    fprintf(ofp,"\n\n\n\t==========================================================================");
    fprintf(ofp,"\n\t========== Assessment Information ==========");
    fprintf(ofp,"\n\t==========================================================================");
    fprintf(ofp,"\n\n\tThe pier natural period is %.2lf seconds. If we",p_info->pier_period);
    fprintf(ofp,"\n\tcompare this to the wave period of %.2lf seconds,",p_info->wave_data->wave_T);

    if((fabs(p_info->pier_period - p_info->wave_data->wave_T)) < 2)
        fprintf(ofp,"\n\tthis IS something to be concerned about...");
    else
        fprintf(ofp,"\n\tthis should be nothing to be concerned about...");

    fprintf(ofp,"\n\n\tThe pile locations can hold the following point loads(lbs)");
    b_temp = p_info->bents;
    p_min = b_temp->pile->vert_load;
```

```
while(b_temp != NULL)
    {
        fprintf(ofp,"\n\n\tBent #%d\n\t", b_temp->id_num);
        p_temp = b_temp->pile;

        while(p_temp != NULL && p_temp->id_let[0] != 'b')
            {
                fprintf(ofp,"  %s: %8.0lf", p_temp->id_let, p_temp->vert_load);

                if(p_min > p_temp->vert_load)
                    p_min = p_temp->vert_load;

                p_temp = p_temp->next;
            }
        b_temp = b_temp->next;
    }

fprintf(ofp,"\n\n\n\tTruck Wheel/Axle Load Information");
fprintf(ofp,"\n\t=========================================");

if(p_min < (2*p_info->loads->hs_limit_wheel_str)||p_min < (2*p_info->loads->h_limit_wheel_str))
    {
        fprintf(ofp,"\n\n\t*****************************************************");
        fprintf(ofp,"\n\t*****************************************************");
        fprintf(ofp,"\n\tWarning!!! There are individual pile load capacities");
        fprintf(ofp,"\n\tthat are less than the computed axle capacities. These");
        fprintf(ofp,"\n\tpiles can NOT support the calculated axle load. Care");
        fprintf(ofp,"\n\tshould be taken that these piles are not overloaded.");
        fprintf(ofp,"\n\t*****************************************************");
        fprintf(ofp,"\n\t*****************************************************");
    }

fprintf(ofp,"\n\n\tFor an HS truck (tractor/trailer) the maximum");
fprintf(ofp,"\n\twheel load is: %.0lf lbs which gives a maximum",p_info->loads->hs_limit_wheel_str);
fprintf(ofp,"\n\taxle load of: %.0lf lbs.",(2*p_info->loads->hs_limit_wheel_str));
fprintf(ofp,"\n\n\tFor an H truck (similar to a tactical 5T) the maximum");
fprintf(ofp,"\n\twheel load is: %.0lf lbs which gives a maximum",p_info->loads->h_limit_wheel_str);
fprintf(ofp,"\n\taxle load of: %.0lf lbs.",(2*p_info->loads->h_limit_wheel_str));
fprintf(ofp,"\n\n\n\tPoint Load Information for Crane Loading");
fprintf(ofp,"\n\t=========================================");

if(p_min < p_info->loads->cap_pt_lim||p_min < p_info->loads->pile_pt_miss)
    {
        fprintf(ofp,"\n\n\t*****************************************************");
        fprintf(ofp,"\n\t*****************************************************");
        fprintf(ofp,"\n\tWarning!!! There are individual pile load capacities");
        fprintf(ofp,"\n\tthat are less than the computed point load capacities.");
        fprintf(ofp,"\n\tThese piles can NOT support the calculated point load.");
        fprintf(ofp,"\n\tCare should be taken that these piles are not overloaded.");
        fprintf(ofp,"\n\t*****************************************************");
        fprintf(ofp,"\n\t*****************************************************");
    }
```

```
fprintf(ofp,"\n\n\tThe pile cap point load capacity when bracketed");
fprintf(ofp,"\n\tby two good piles is: %.0lf lbs.",p_info->loads->cap_pt_lim);
fprintf(ofp,"\n\n\tFor a pile cap over a severely damaged or missing");
fprintf(ofp,"\n\tpile, the point load capacity is: %.0lf lbs.",p_info->loads->pile_pt_miss);
fprintf(ofp,"\n\n\n\tForklift Assessment");
fprintf(ofp,"\n\t==========");
fprintf(ofp,"\n\n\tThe Y or N indicates the stringers ability to handle");
fprintf(ofp,"\n\tthe forklift axle loading. However, if an individual pile");
fprintf(ofp,"\n\tload capacity is less than the indicate axle capacity, the");
fprintf(ofp,"\n\tforklift will NOT be able to operate safely over those piles.");
fprintf(ofp,"\n\n\t\t  5T Forklift: %c,   axle: 20,000",p_info->loads->fork_lim_5);
fprintf(ofp,"\n\t\t  6T Forklift: %c,   axle: 23,000",p_info->loads->fork_lim_6);
fprintf(ofp,"\n\t\t7.5T Forklift: %c,   axle: 19,000",p_info->loads->fork_lim_75);
fprintf(ofp,"\n\t\t  8T Forklift: %c,   axle: 30,500",p_info->loads->fork_lim_8);
fprintf(ofp,"\n\t\t 10T Forklift: %c,   axle: 35,000",p_info->loads->fork_lim_10);
fprintf(ofp,"\n\t\t 12T Forklift: %c,   axle: 44,300",p_info->loads->fork_lim_12);
fprintf(ofp,"\n\t\t 15T Forklift: %c,   axle: 58,000",p_info->loads->fork_lim_15);
fprintf(ofp,"\n\t\t 20T Forklift: %c,   axle: 98,000",p_info->loads->fork_lim_20);

uni_min = p_info->loads->string_uni_lim_psi;

if(uni_min > p_info->loads->cap_uni_lim_psi)
   uni_min = p_info->loads->cap_uni_lim_psi;

if(uni_min > p_info->loads->plank_uni_lim_psi)
   uni_min = p_info->loads->plank_uni_lim_psi;

fprintf(ofp,"\n\n\tThe Uniform Load Capacity is: %.2lf psi or %.2lf psf.",uni_min,(144*uni_min));

if(uni_min > p_info->loads->pile_uni_min)
   {
      fprintf(ofp,"\n\n\tHowever, some areas are even lower as follows (psi)");
      b_temp = p_info->bents;

      while(b_temp != NULL)
         {
            fprintf(ofp,"\n\n\tBent #%d", b_temp->id_num);
            p_temp = b_temp->pile;

            while(p_temp != NULL && p_temp->id_let[0] != 'b')
               {
                  p_load = uni_min;

                  if(p_temp->uni_load_psi < p_load)
                     p_load = p_temp->uni_load_psi;

                  fprintf(ofp,"  %s: %7.2lf", p_temp->id_let, p_load);
                  p_temp = p_temp->next;
               }
            b_temp = b_temp->next;
         }
}
```

```
fprintf(ofp,"\n\n\tFor properly loaded containers, the following");
fprintf(ofp,"\n\tuniform load values (psf) are applicable:");
fprintf(ofp,"\n\n\t\t          # of containers in stack");
fprintf(ofp,"\n\n\t\t L   W   H       1    2    3");
fprintf(ofp,"\n\n\t\t═══════════════════════════════════════════════");
fprintf(ofp,"\n\n\t\t40.0x8.0x8.0      211   422   633");
fprintf(ofp,"\n\t\t29.9x8.0x8.0      234   468   702");
fprintf(ofp,"\n\t\t19.9x8.0x8.0      284   568   852");
fprintf(ofp,"\n\t\t 9.8x8.0x8.0      286   572   858");
fprintf(ofp,"\n\t\t 6.4x8.0x8.0      305   610   915");
fprintf(ofp,"\n\t\t 4.8x8.0x8.0      294   589   883");
fprintf(ofp,"\n\n\n\tShip Information");
fprintf(ofp,"\n\t═══════════════");

if(p_info->ship == 1||p_info->ship == 2)
   {
      fprintf(ofp,"\n\n\tThe following ship information was input:");
      fprintf(ofp,"\n\tShip length: %.0lf ft", p_info->ship_data->length);
      fprintf(ofp,"\n\tShip draft: %.0lf ft", p_info->ship_data->draft);
      fprintf(ofp,"\n\tShip equivalent freeboard: %.0lf ft", p_info->ship_data->freeboard);

      if(p_info->ship == 1)
         {
            fprintf(ofp,"\n\tUsing the information provided, it is NOT safe for");
            fprintf(ofp,"\n\tthis ship to moor at this pier.");

            if(p_info->est_ton_ship > 0)
               {
                  fprintf(ofp,"\n\tYou should be able to berth a ship of");
                  fprintf(ofp,"\n\tapproximately %.2lf long tons.",(0.8*p_info->est_ton_ship));
               }
         }
      else
         {
            fprintf(ofp,"\n\tUsing the information provided, it is safe for");
            fprintf(ofp,"\n\tthis ship to moor at this pier.");
         }
   }

fprintf(ofp,"\n\n\tWhen performing load calculations, the following wind");
fprintf(ofp,"\n\tand current forces should be applied to the area of the");
fprintf(ofp,"\n\tship under consideration:");

if(p_info->ship > 0)
   {
      w_load = 1.3*p_info->ship_data->C1*p_info->ship_data->C2*fabs(p_info->forces->wind_x);
      fprintf(ofp,"\n\n\tWind Loading: %.2lf psf perpendicular to the pier",w_load);
      w_load = 1.3*p_info->ship_data->C1*p_info->ship_data->C2*fabs(p_info->forces->wind_70);
      fprintf(ofp,"\n\n\tFor winds of 70mph, this force would be: %.2lf psf.",w_load);
   }
else
   {
      if(fabs(p_info->forces->wind_x) > 0.0)
```

```
            {
                fprintf(ofp,"\n\n\tWind Loading is approximately: %.2lf psf perpendicular to the
pier",fabs(p_info->forces->wind_x));
                fprintf(ofp,"\n\n\tFor winds of 70mph, this force would be approximately: %.2lf
psf",fabs(p_info->forces->wind_70));
            }
        else
            fprintf(ofp,"\n\n\tThere is no Wind Loading with the given input.");
    }

    if(fabs(p_info->forces->current_ship_x) > 0.0)
        {
            fprintf(ofp,"\n\n\tCurrent Loading: %.2lf psf perpendicular to the pier",fabs(p_info->forces-
>current_ship_x));
        }
    else
        fprintf(ofp,"\n\n\tThere is no Current Loading with the given input.");

    fprintf(ofp, "\n%c", '\0');
    fclose(ofp);
    return(f_temp);
}

/*************************************************************/
void prn_to_screen(void)
{
    char choice, y_n, f_temp, fname[13] = "pier_in0.dat";
    int flag = 0, flag1 = 1;
    int handle;
    int bytes;
    void *buf;

    while(flag == 0)
        {
            clean();
            printf("\n\n\t\tRSAP - Rapid Structural Assessment, Pier v1.0");
            printf("\n\n\t\t\tDisplay File Menu");
            printf("\n\n\t\t\t1)View file just entered");
            printf("\n\t\t\t2)View previously saved file");
            printf("\n\t\t\t3)Return to Main Menu");
            printf("\n\n\t\t\tEnter selection: ");
            choice = getche();

            switch(choice)
                {
                    case '1':
                        fname[7] = 't';

                        if ((handle = open(fname, O_RDONLY | O_TEXT, S_IREAD)) == -1)
                            {
                                fflush(stdin);
                                clrscr();
                                printf("\n\n\n\t\tYou need to input data first!!!\n\t");
```

```
            pause();
        }
    else
        {
            fflush(stdin);
            clrscr();
        if((buf = calloc(1, filelength(handle))) == NULL)
            {
                printf("\n\tError opening read buffer");
                exit(-1);
            }
            bytes = read(handle, buf, filelength(handle));
            handle = close(handle);
            printf("%s", buf);
            free(buf);
            pause();
        }
            break;

case '2':
    printf("\n\n\t\tEnter a number between 0-9. You should have input");
    printf("\n\t\tthis number when saving the user data file.");
    printf("\n\n\t\tPlease enter number: ");
    f_temp = (int)(check_num()+48);
    fname[7] = f_temp;
    flag1 = 1;

    while(flag1 == 1)
        {
            if ((handle = open(fname, O_RDONLY | O_TEXT, S_IREAD)) == -1)
                {
                    printf("\n\n\tFile not found!!! Do you wish to try");
                    printf(" another file (y/n)? ");
                    y_n = getche();

                    while(y_n != 'y'&&y_n != 'Y'&&y_n != 'n'&&y_n != 'N')
                        {
                            prt_inv();
                            printf("\n\tDo you wish try another file (y/n)?  ");
                            y_n = getche();
                        }

                    if(y_n == 'y'||y_n == 'Y')
                        {
                            printf("\n\n\tPlease enter new number: ");
                            f_temp = (int)(check_num()+48);
                            fname[7] = f_temp;
                        }
                    else
                        flag1 = 2;
                }
            else
                flag1 = 0;
```

```
                    }
                if(flag1 == 0)
                    {
                        fflush(stdin);
                        clrscr();
                        if((buf = calloc(1, filelength(handle))) == NULL)
                            {
                                printf("\n\tError opening read buffer");
                                exit(-1);
                            }
                        bytes = read(handle, buf, filelength(handle));
                        handle = close(handle);
                        printf("%s", buf);
                        free(buf);
                        pause();
                    }
                    break;

                case '3':
                    flag = 1;
                    break;

                default:
                    flag = 0;
                    prt_inv();
                    break;
            }
        }
}

/*********************************************************/
void pile_plot(PierData *p_info, char f_temp)
{
    FILE *ofp,*tfp;
    Bent *b_temp;
    Pile *p_temp;
    char fname[11] = "pil_plt0.m";
    char tname[11] = "uni_plt0.m";
    double uni_min, p_load;

    time_t t;

    time(&t);
    fname[7] = f_temp;
    ofp = fopen(fname, "w");

    if(ofp == NULL)
        {
            printf("\nERROR: Cannot open user output file!\n");
            exit(-1);
        }

    fprintf(ofp,"%% ");
```

```
date_plot(ofp);
fprintf(ofp,"%%");
fprintf(ofp,"\n%% This file processes the output of RSAP to plot the");
fprintf(ofp,"\n%% strength contours of the pile capacities.");
fprintf(ofp,"\n%%");
fprintf(ofp,"\n%% (C)1999 - All rights reserved");
fprintf(ofp,"\n%%          R. J. Keiter(rjkeiter@iname.com)");
fprintf(ofp,"\n\nbents = %d;", p_info->no_of_bents);
fprintf(ofp,"\npiles = %d;", p_info->piles_per_bent);
fprintf(ofp,"\nb_span = %.2lf;", p_info->bent_spread);
fprintf(ofp,"\np_span = %.2lf;", p_info->pile_spread);
fprintf(ofp,"\n");
fprintf(ofp,"\n%%z = capacity matrix");
fprintf(ofp,"\n");
fprintf(ofp,"\nz = [");
b_temp = p_info->bents;

while(b_temp != NULL)
    {
        fprintf(ofp,"[");
        p_temp = b_temp->pile;

        while(p_temp != NULL && p_temp->id_let[0] != 'b')
            {
                fprintf(ofp,"%.0lf", p_temp->vert_load);
                p_temp = p_temp->next;

                if(p_temp != NULL && p_temp->id_let[0] != 'b')
                    fprintf(ofp," ");
            }

        b_temp = b_temp->next;

        if(b_temp == NULL)
            fprintf(ofp,"]];\n");
        else
            {
                fprintf(ofp,"]\n");
                fprintf(ofp,"     ");
            }
    }

fprintf(ofp,"\nx_lim = (piles - 1) * p_span;");
fprintf(ofp,"\nx_lim2 = x_lim/2;");
fprintf(ofp,"\ny_lim = (bents - 1) * b_span;");
fprintf(ofp,"\n");
fprintf(ofp,"\nx=-x_lim2:p_span:x_lim2;");
fprintf(ofp,"\ny=0:b_span:y_lim;");
fprintf(ofp,"\n");
fprintf(ofp,"\nxi = linspace(-x_lim2,x_lim2,x_lim);");
fprintf(ofp,"\nyi = linspace(0,y_lim,y_lim/2);");
fprintf(ofp,"\n");
fprintf(ofp,"\n[X,Y]=meshgrid(x,y);");
```

```
fprintf(ofp,"\n[Xi,Yi] = meshgrid(xi,yi);");
fprintf(ofp,"\n");
fprintf(ofp,"\nZ = interp2(x,y,z,Xi,Yi,'cubic');");
fprintf(ofp,"\n");
fprintf(ofp,"\ncolormap(gray);");
fprintf(ofp,"\npcolor(Xi,Yi,Z);");
fprintf(ofp,"\ncolorbar('v');");
fprintf(ofp,"\n");
fprintf(ofp,"\nshading interp");
fprintf(ofp,"\nhold on");
fprintf(ofp,"\ncontour(Xi,Yi,Z,10,'k')");
fprintf(ofp,"\n");
fprintf(ofp,"\naxis([(-x_lim * 2) (x_lim * 2) 0 (y_lim + y_lim/4)]);");
fprintf(ofp,"\n");
fprintf(ofp,"\nyl=linspace(0,y_lim,2);");
fprintf(ofp,"\nxl=linspace(-x_lim2,-x_lim2,2);");
fprintf(ofp,"\nplot(xl,yl);");
fprintf(ofp,"\n");
fprintf(ofp,"\nyt=linspace(y_lim,y_lim,2);");
fprintf(ofp,"\nxt=linspace(-x_lim2,x_lim2,2);");
fprintf(ofp,"\nplot(xt,yt);");
fprintf(ofp,"\n");
fprintf(ofp,"\nxr=linspace(x_lim2,x_lim2,2);");
fprintf(ofp,"\nplot(xr,yl);");
fprintf(ofp,"\ntitle('Plot of Point Loading Capacity(lbs) - Based upon Pile Locations');");
fprintf(ofp,"\nxlabel('Pier Width - ft');");
fprintf(ofp,"\nylabel('Pier Length - ft');");
fclose(ofp);

uni_min = p_info->loads->string_uni_lim_psi;
tname[7] = f_temp;
tfp = fopen(tname, "w");

if(tfp == NULL)
    {
        printf("\nERROR: Cannot open user output file!\n");
        exit(-1);
    }

fprintf(tfp,"%% ");
date_plot(tfp);
fprintf(tfp,"%%");
fprintf(tfp,"\n%% This file processes the output of RSAP to plot the");
fprintf(tfp,"\n%% uniform loading contours of the pier.");
fprintf(tfp,"\n%%");
fprintf(tfp,"\n%% (C)1999 - All rights reserved");
fprintf(tfp,"\n%%          R. J. Keiter(rjkeiter@iname.com)");
fprintf(tfp,"\n\nbents = %d;", p_info->no_of_bents);
fprintf(tfp,"\npiles = %d;", p_info->piles_per_bent);
fprintf(tfp,"\nb_span = %.2lf;", p_info->bent_spread);
fprintf(tfp,"\np_span = %.2lf;", p_info->pile_spread);
fprintf(tfp,"\n");
fprintf(tfp,"\n%%z = capacity matrix");
```

```
fprintf(tfp,"\n");
fprintf(tfp,"\nz = [");
b_temp = p_info->bents;

while(b_temp != NULL)
   {
      fprintf(tfp,"[");
      p_temp = b_temp->pile;

      while(p_temp != NULL && p_temp->id_let[0] != 'b')
         {
            p_load = uni_min;

            if(p_temp->uni_load_psi < p_load)
               p_load = p_temp->uni_load_psi;

            fprintf(tfp,"%.2lf",p_load);
            p_temp = p_temp->next;

            if(p_temp != NULL && p_temp->id_let[0] != 'b')
               fprintf(tfp," ");
         }

      b_temp = b_temp->next;

      if(b_temp == NULL)
         fprintf(tfp,"]];\n");
      else
         {
            fprintf(tfp,"]\n");
            fprintf(tfp,"    ");
         }
   }

fprintf(tfp,"\nx_lim = (piles - 1) * p_span;");
fprintf(tfp,"\nx_lim2 = x_lim/2;");
fprintf(tfp,"\ny_lim = (bents - 1) * b_span;");
fprintf(tfp,"\n");
fprintf(tfp,"\nx=-x_lim2:p_span:x_lim2;");
fprintf(tfp,"\ny=0:b_span:y_lim;");
fprintf(tfp,"\n");
fprintf(tfp,"\nxi = linspace(-x_lim2,x_lim2,x_lim);");
fprintf(tfp,"\nyi = linspace(0,y_lim,y_lim/2);");
fprintf(tfp,"\n");
fprintf(tfp,"\n[X,Y]=meshgrid(x,y);");
fprintf(tfp,"\n[Xi,Yi] = meshgrid(xi,yi);");
fprintf(tfp,"\n");
fprintf(tfp,"\nZ = interp2(x,y,z,Xi,Yi,'cubic');");
fprintf(tfp,"\n");
fprintf(tfp,"\ncolormap(gray);");
fprintf(tfp,"\npcolor(Xi,Yi,Z);");
fprintf(tfp,"\ncolorbar('v');");
fprintf(tfp,"\n");
```

```
    fprintf(tfp,"\nshading interp");
    fprintf(tfp,"\nhold on");
    fprintf(tfp,"\ncontour(Xi,Yi,Z,10,'k')");
    fprintf(tfp,"\n");
    fprintf(tfp,"\naxis([(-x_lim * 2) (x_lim * 2) 0 (y_lim + y_lim/4)]);");
    fprintf(tfp,"\n");
    fprintf(tfp,"\nyl=linspace(0,y_lim,2);");
    fprintf(tfp,"\nxl=linspace(-x_lim2,-x_lim2,2);");
    fprintf(tfp,"\nplot(xl,yl);");
    fprintf(tfp,"\n");
    fprintf(tfp,"\nyt=linspace(y_lim,y_lim,2);");
    fprintf(tfp,"\nxt=linspace(-x_lim2,x_lim2,2);");
    fprintf(tfp,"\nplot(xt,yt);");
    fprintf(tfp,"\n");
    fprintf(tfp,"\nxr=linspace(x_lim2,x_lim2,2);");
    fprintf(tfp,"\nplot(xr,yl);");
    fprintf(tfp,"\ntitle('Plot of Uniform Loading Capacity(psi)');");
    fprintf(tfp,"\nxlabel('Pier Width - ft');");
    fprintf(tfp,"\nylabel('Pier Length - ft');");
    fclose(tfp);
}

/*********************************************************/
double test_input_double(double check)
{
    char in_value[20] = {0}, choice;
    int ctr = 0, j;
    double i = 0.0;
    int flag = 0;

    while(flag == 0)
        {
            scanf(" %s", in_value);

            if(in_value[0] == '0'&&strlen(in_value) == 1&&check == 0.0)
                return(0.0);

            i = atof(in_value);

            for(j = 0; j < strlen(in_value); j++)
                {
                    if(isalpha(in_value[j]))
                        i = -1.0;
                }

            if(i < check)
                {
                    ctr++;

                    if(ctr >= 3)
                        {
                            printf("\n\tThat's 3 incorrect inputs in a row.");
                            printf("\n\tDo you wish to quit(y/n)? ");
```

```
                    choice = getche();

                    if(choice == 'y'||choice == 'Y')
                        {
                            printf("\n\n\tLater!!!");
                            exit(0);
                        }
                    else
                        {
                            ctr = 0;
                        }
                }
            prt_inv();
            printf("\n\tRe-enter value: ");
            in_value[0] = '\0';
        }
    else
        flag = 1;
}

return(i);
}

/*********************************************************/
int test_input_int(int check)
{
    char in_value[20] = {0}, choice;
    int i = 0, ctr = 0, flag = 0, j;

    while(flag == 0)
        {
            scanf(" %s", in_value);

            if(in_value[0] == '0'&&strlen(in_value) == 1&&check == 0)
                return(0);

            i = atoi(in_value);

            for(j = 0; j < strlen(in_value); j++)
                {
                    if(isalpha(in_value[j]))
                        i = -1.0;
                }

            if(i < check)
                {
                    ctr++;

                    if(ctr >= 3)
                        {
                            printf("\n\tThat's 3 incorrect inputs in a row.");
                            printf("\n\tDo you wish to quit(y/n)? ");
                            choice = getche();
```

```
                    if(choice == 'y'||choice == 'Y')
                        {
                            printf("\n\n\tLater!!!");
                            exit(0);
                        }
                    else
                        {
                            ctr = 0;
                        }
                }
            prt_inv();
            printf("\n\tRe-enter value: ");
            in_value[0] = '\0';
        }
    else
        flag = 1;
    }

  return(i);
}

/*************************************************************/
int check_num(void)
{
  char temp[5];
  int i = 10;

  while(i > 9)
    {
      scanf(" %s", &temp);
      i = atoi(temp);

      if((i > 0&&i < 10)||(temp[0] == '0'&&strlen(temp) == 1))
        {
          return(i);
        }
      else
        {
          prt_inv();
          printf("\n\t\tPlease enter number: ");
          i = 10;
        }
    }

  return(0);
}

/*************************************************************/
void clean(void)
{
  clrscr();
  printf("\n\n\t\t(C)1999 - All rights reserved.");
```

```
    printf("\n\t\t        R. J. Keiter (rjkeiter@iname.com)");
    printf("\n\n\t\t        Today is: ");
    put_date();
    printf("\n\n");
}

/*************************************************************/
void date_plot(FILE *ofp)
{
    time_t t;

    time(&t);
    fprintf(ofp,"%s", ctime(&t));
}

/*************************************************************/
void pause(void)
{
    char msg[27] = {"Press enter to continue..."};

    printf("\n\t%s", msg);
    while ((getchar()) != '\n') {}
    fflush(stdin);
}

/*************************************************************/
void put_date(void)
{
    struct date d;

    getdate(&d);
    printf("%d/%d/%d", d.da_mon, d.da_day, d.da_year);
}

/*************************************************************/
void prt_err(char temp[35])
{
    printf("\nERROR: Cannot create %s Structure!\n", temp);
    exit(-1);
}

/*************************************************************/
void prt_inv(void)
{
 fflush(stdin);
    printf("\n\n\tInvalid input! Please try again...\n\n");
    pause();
 clrscr();
}

/*************************************************************/
```

Appendix F - Sample RSAP Output Data File

File generated: Fri Jan 15 05:27:06 1999

The following information was input in to RSAP.
(C)1999 - All rights reserved. R. J. Keiter

The wind speed was measured at 15 mph, 30' above
the water's surface at an angle of 35 degrees.
The current speed was measured at 0.68 mph at an
angle of 85 degrees.
1.17' waves at 14 seconds were observed in 35' of water at an
angle of 10 degrees.

****
********** ATTENTION!!! **********
****

These waves do not fall under the theory used here to
analyze the forces due to waves. The actual forces will
most likely be larger than that calculated and used in
this assessment. Proceed with caution!

****
********** ATTENTION!!! **********
****

Soil Type: Medium Sand

This analysis was performed using a factor of safety of: 2.00

Number of bents: 7
The distance between bents is: 12.00'
The pilings are 14" Douglas Fir with 5" of bio-fouling.
Number of piles per bent: 5
Distance between pilings: 8.00'

The superstructure is constructed of White Oak,
has cross bracing, and has a solid type pile cap.
The pile cap member dimensions are 14" x 20".
The stringers are 3" x 16" and are centered 1.50' apart.
The deck planking is 12" x 3".

The piling inspection information is as follows:
(The number following the bent number is the distance from the bottom to where the piling meets the cross bracing.)

Bent #1 - 25'
A: ND B: ND C: ND D: ND E: ND

Bent #2 - 30'
A: ND B: MN C: ND D: ND E: ND

Bent #3 - 40'
A: ND B: ND C: ND D: SV E: ND

Bent #4 - 40'
A: ND B: ND C: ND D: ND E: ND

Bent #5 - 40'
A: ND B: ND C: SV D: ND E: ND

Bent #6 - 40'
A: ND B: MJ C: ND D: ND E: ND

Bent #7 - 40'
A: ND B: ND C: ND D: ND E: ND

==========

========== Assessment Information ==========

==========

The pier natural period is 0.56 seconds. If we compare this to the wave period of 14.00 seconds, this should be nothing to be concerned about...

The pile locations can hold the following point loads(lbs)

Bent #1
A: 206048 B: 204741 C: 204741 D: 204741 E: 206048

Bent #2
A: 206048 B: 149607 C: 204741 D: 204741 E: 206048

Bent #3

A: 206048 B: 204741 C: 204741 D: 0 E: 206048

Bent #4
A: 206048 B: 204741 C: 204741 D: 204741 E: 206048

Bent #5
A: 206048 B: 204741 C: 0 D: 204741 E: 206048

Bent #6
A: 206048 B: 0 C: 204741 D: 204741 E: 206048

Bent #7
A: 206048 B: 204741 C: 204741 D: 204741 E: 206048

Truck Wheel/Axle Load Information

****
****
Warning!!! There are individual pile load capacities
that are less than the computed axle capacities. These
piles can NOT support the calculated axle load. Care
should be taken that these piles are not overloaded.
****
****

For an HS truck (tractor/trailer) the maximum
wheel load is: 13274 lbs which gives a maximum
axle load of: 26548 lbs.

For an H truck (similar to a tactical 5T) the maximum
wheel load is: 13274 lbs which gives a maximum
axle load of: 26548 lbs.

Point Load Information for Crane Loading

****
****
Warning!!! There are individual pile load capacities
that are less than the computed point load capacities.
These piles can NOT support the calculated point load.

Care should be taken that these piles are not overloaded.
****
****

The pile cap point load capacity when bracketed
by two good piles is: 171802 lbs.

For a pile cap over a severely damaged or missing
pile, the point load capacity is: 52013 lbs.

Forklift Assessment
==========

The Y or N indicates the stringers ability to handle
the forklift axle loading. However, if an individual pile
load capacity is less than the indicate axle capacity, the
forklift will NOT be able to operate safely over those piles.

5T Forklift: Y, axle: 20,000
6T Forklift: Y, axle: 23,000
7.5T Forklift: N, axle: 19,000
8T Forklift: N, axle: 30,500
10T Forklift: N, axle: 35,000
12T Forklift: N, axle: 44,300
15T Forklift: N, axle: 58,000
20T Forklift: N, axle: 98,000

The Uniform Load Capacity is: 10.24 psi or 1474.90 psf.

However, some areas are even lower as follows (psi)

Bent #1 A: 10.24 B: 10.24 C: 10.24 D: 10.24 E: 10.24

Bent #2 A: 10.24 B: 10.24 C: 10.24 D: 10.24 E: 10.24

Bent #3 A: 10.24 B: 10.24 C: 10.24 D: 0.00 E: 10.24

Bent #4 A: 10.24 B: 10.24 C: 10.24 D: 10.24 E: 10.24

Bent #5 A: 10.24 B: 10.24 C: 0.00 D: 10.24 E: 10.24

Bent #6 A: 10.24 B: 0.00 C: 10.24 D: 10.24 E: 10.24

Bent #7 A: 10.24 B: 10.24 C: 10.24 D: 10.24 E: 10.24

For properly loaded containers, the following uniform load values (psf) are applicable:

	# of containers in stack		
L W H	1	2	3
40.0x8.0x8.0	211	422	633
29.9x8.0x8.0	234	468	702
19.9x8.0x8.0	284	568	852
9.8x8.0x8.0	286	572	858
6.4x8.0x8.0	305	610	915
4.8x8.0x8.0	294	589	883

Ship Information

When performing load calculations, the following wind and current forces should be applied to the area of the ship under consideration:

Wind Loading is approximately: 0.32 psf perpendicular to the pier

For winds of 70mph, this force would be approximately: 7.19 psf

Current Loading: 0.99 psf perpendicular to the pier